12th

ARRL DIGITAL COMMUNICATIONS CONFERENCE

TAMPA, FLORIDA
SEPTEMBER 10-11, 1993

12th ARRL DIGITAL COMMUNICATIONS CONFERENCE

HOSTS: The Tampa Local Area Network (TPALAN)

Printed in USA

$12.00 in USA

ISBN: O-87259-442-4

First Edition

Foreword

The American Radio Relay League takes pride in publishing these papers of the 12th ARRL Digital Communications Conference (formerly the Amateur Radio Computer Networking Conference).

This year's papers include numerous applications of packet technology, network improvement schemes and technical developments. Network management and monitoring continue to grow in sophistication, demonstrated by papers about using TCP/IP over the ROSE network, network routing, and network traffic monitoring. Information gathering and dissemination via digital networks seems to be one of the de facto themes of this year's conference, as exemplified by papers on disaster communications, network information services and mail distribution. A new TNC design, a high-speed 23-cm system, and digital signal processing applied to packet are among the technical developments. Improvements to older techniques, such as AMTOR and a software-approach packet system are included, too.

As in previous conferences, all papers in these proceedings are unedited and solely the work of the authors.

David Sumner, K1ZZ
Executive Vice President

Newington, Connecticut
September, 1993

Table of Contents

USING AIRBORNE DIGITAL REPEATERS IN EMERGENCY COMMUNICATIONS

A paper prepared for the 12th annual
ARRL Networking Conference

September 10-11, 1993
Tampa, Florida

. Prepared by Gary Arnold, WB2WPA @ WB2WPA
3200 Areca Avenue
Naples, FL 33962-5912

The state of Florida is, as pointed out so dramatically on August 23-224, 1992, vulnerable to the destruction of a hurricane. Communications links so vital to recovery are often among the weakest, most vulnerable targets. Linking the southern half of the state with the state capitol in Tallahassee is more than difficult, because of the distances involved (Figure 1).

On the southwest coast of Florida in Naples, 110 miles due west of Ft. Lauderdale, the Collier County Emergency Operations Center provides link from EOCs up the west coast to Dade, Broward and Monroe Counties, including the National Hurricane Center in Coral Gables, Florida. A dual port network node is located at Ochopee providing 220 and 440 links to the east coast. Another is planned for a federal government site near the Dade County line, providing redundant sites along US 41 between Naples and Miami.

To the north, the FMY node stack is located at the Lee County EOC in Ft. Myers, consisting of 2 meters, 220 and 440 nodes some 40 miles north of Naples. Thirty miles to the north is Punta Gorda with a 2 meter/440 node serving Charlotte County, and linking to Sarasota. From there, the ROSE network takes over to Tampa,. and Pasco County. There, the northbound network virtually stops, except for a tenuous connection along the I-4 corridor (Figure 2).

Packet radio connections have been made between the WB2WPA BBS in Naples, and the state EOC in Tallahassee, but the connections have relied on enhanced paths to provide a slow link which is not capable of providing reliable extended periods of communication. If a storm was to destroy the terrestrial VHF/HF network, the entire state would have to rely on HF packet communications to link the southern half of the state with Tallahassee.

At the 1989 Florida Governor% Hurricane Conference in Tampa, I proposed a method of flying nodes in "orbits" along the length of the 120 mile wide peninsula at relatively high altitudes, providing a 2 hop link between the southern end of the 'state and the State EOC.

Under the plan, Civil Air Patrol aircraft would fly circles between 6,000 and 10,000 feet near Wauchula in Highlands County, some 50 miles ESE of Tampa Bay, and near Cedar Key or Gainesville in the north central portion of the state. Conceivably, these two aircraft could link the state from Jacksonville and Tallahassee to Miami and Naples. A third aircraft over the panhandle near Apalachicola would tie in the rest of the state.

Basic tests were run using a single aircraft to basically determine the maximum usable range of air to ground link. A Cessna 172 was equipped with a small lantern battery, a Yaesu 5 watt handheld, and a PacCom TNC carried in the back seat of the aircraft. For the original test, a quarter wave rubber duck" antenna was strapped to an unused VHF antenna on the belly of the aircraft.

An operator at the keyboard 'of the WB2WPA-5 BBS in Naples established a contact with the aircraft before takeoff (the airport is about 2 miles from the EOC, and the antenna is at 140 feet). A CAP operator in the EOC was to maintain voice contact with the aircraft and CAP HQ on CAP VHF frequencies.

The aircraft departed Naples on a heading of 045 degrees, and was directed to fly to LaBelle in Hendry County, about 50 miles north-northeast of Naples, while climbing to 6,000 feet. At the EOC, an additional 2 meter radio monitored the packet frequency in use, 145.050. The frequency was selected because it was the Naples LAN frequency and local operators were encouraged to access the aircraft. The frequency is also used as a LAN frequency in the Tampa Bay area on the west coast, and Stuart and Daytona Beach on the east coast.

Over the LaBelle VOR the aircraft was full quieting,both on packet and voice. Connects were made and broken repeatedly to be sure that they could be made when needed, covering both coasts and virtually the entire southern half ot the state. The aircraft was also used to connect through a landbased node to the Daytona Beach area (Figure 3). During connections, messages were up to and downloaded from the PBBS in the TNC. Heard lists were downloaded to determine how well the aircraft was hearing.

From LaBelle, the aircraft was sent north toward Wauchula, maintaining 6,000 feet. As it continued north, Key West became less and less apparent, but more and more stations were heard in the Lakeland (30 miles east of Tampa) and Daytona Beach areas.

From Wauchula, contacts were made between Naples, Tampa and Daytona with ease, until the aircraft reached its northernmost point in Hardee County (Figure 4). Signals were still usable, but collisions on such a busy frequency were slowing things down.

The aircraft returned to Naples, and continued providing excellent coverage of the southern half of the state, almost to touchdown.

This intial experiment indicated that at-6,000 feet, we could expect good communications from the 140 foot high omni antenna at the EOC to the rubber duck antenna on the plane to about 110420 miles. Some polarization problems were seen as the antenna flexed toward the rear of the aircraft, and plans were then discussed aimed at retrying the experiment at other altitudes, and with a more rigid, permanently mounted antenna.

In March of 1993, the Florida Wing of Civil Air Patrol held its annual Disaster Recovery Exercize (DREX) utilizing several bases around the state, and the state EOC in Tallahassee.

In Naples, it was decided to include some testing and training using the County's new $125,000 Mobile Command Center, which is equipped with packet using both HF and VHF. Antennas are roof and tower mounted, with power levels of 5 and 40 watts available on HF packet.

CAP frequencies would be used, and two aircraft would be available carrying packet digis, but not nodes, as CAP is currently using digipeater technology instead of transport level nodes. One aircraft would fly from Naples and head north. A second would leave the Tampa Bay area and head up the coast.

Again, voice communications were available for coordination with the aircraft on VHF, as well as HF CAP frequencies for coordination with other CAP bases and Tallahassee. FEMA's Operation Secure frequencies were also used direct to Tallahassee and the State EOC.

The equipment on both aircraft was basically the same as in the original test, except that permanently mounted quarter wave rigid belly antennas were installed. This kept the packet antennas away from the CAP and aircraft VHF antennas on top of the cabin.

Because of poor performance and highly irregular connections between Tallahassee and the southern half of the state using the VHF/UHF network, state officials were skeptical at best, As the exercize continued, the aircraft, flying at 10,000 feet, moved toward there target areas.

Again, connections were made and broken repeatedly to test "connectability? Heard lists were dumped, and the Naples. aircraft was hearing the Tampa aircraft very well. An operator on board confirmed good signals. .

From Naples a connection was requested of the Tampa aircraft TNC, via the Naples aircraft. Even though it was a digied connection, it was suprisingly fast. The heard list did not indicate that the state EOC was being heard. The northern aircraft was contacted via an HF request to its base. Its position was northern Hillsborough County at 10,000 feet. A request was sent to move farther north, as the heard list was downloaded repeatedly, looking for Tallahassee.

On HF, contact was maintained with the State EOG, which continued sending connect requests to the northernmost aircraft. Finally, a connect was reported. Both Naples and Tallahassee were connected to the same airborn TNC, Tallahassee directly, and Naples via a'one digi hop!

The mission commander in Naples appeared in the bus, asking if we could send a priority message to Tallahassee. The message was created using a text editor in the onboard computer in the Command Center. Connections with the aircraft were broken from both ends. Coordinating on HF, the command center requested a connection with Tallahassee via the Naples aircraft and the Tampa aircraft, now orbiting near New Port Richey in Pasco County...and , running a little low on fuel!

The Tampa aircraft was a bit farther 'south than had been hoped, right on the edge of capability of maintaining communications with the State. The connection went through! The first south Florida to Tallahassee packet connection via aircraft was made.

The text was sent, in formal CAP format. When the STA light went out (finally) in the Command Center, we knew it had gone through. A few seconds later came an HF voice confirmation: they HAD the message in Tallahassee!

A second message had been prepared, and the connection was broken, in an attempt to upload the message to the second aircraft's PBBS. It was loaded fairly easily, again digipeating through the first aircraft, which was now also running a bit low on fuel. From Naples, on HF, the Tallahassee station was advised that the message had been uploaded to the Tampa aircraft PBBS, and that we had been told that the aircraft was returning to Tampa.

Sadly, the aircraft had moved too far south and had begun its descent into Tampa by then, and Tallahassee was not able t0 retrieve the message. The Naples plane was ordered home, and the experiment was concluded.

Plans were immediately discussed for a third experiment, this time flying amateur radio NET/ROM or TheNet nodes, or ROSE switches aboard the aircraft, in order to get TNC to TNC acknowledgement at each step improving throughput. We are also looking at using discrete VHF/UHF frequencies and higher speeds to allow more traffic to pass through the system. Some examples for future tests:

1: Fly single frequency nodes at 1200 baud to link north and south Florida on otherwise quiet frequencies.

2: Equip the aircraft with 9600 baud or faster TNCs.

3: Using sophisticated laptop computers, attempt flying full service BBS software, allowing land based BBS hubs to autoforward to aircraft 1, which would then forward to aircraft 2 which would then downlink to the north Florida hub BBS.

4: Develop cooperative plans between CAP and ARES/RACES to permanently equip at least 1 plane in Naples, Miami, Tampa, Orlando, Gainesville, Jacksonville, Tallahassee and the panhandle with high speed links to the ground and each other, capable of being quickly deployed in a rotation that will keep three aircraft in the air for as long as possible, or on a scheduled basis providing at least interim packet communications using both amateur and CAP frequencies linking emergency operations centers throughout the state.

5: Locate and assign packet equipped mobile communications centers for deployment along with amateur radio communications teams into disaster areas in coordination with the Florida ARES/RACES Communications Assistance Team Plan adopted in July 1993.

6: Attempt to connect the entire state of Florida using 3 aircraft (Figure 5) by flying a third aircraft north and slightly west of Apalachicola in the panhandle. If further linking were needed, say to FEMA in Atlanta, a fourth aircraft in Georgia might well provide that link.

Further testing is planned with these experiments in mind. We would be most interested in hearing from other groups who have done airborne testing, or who would be interested in assisting in our further tests.

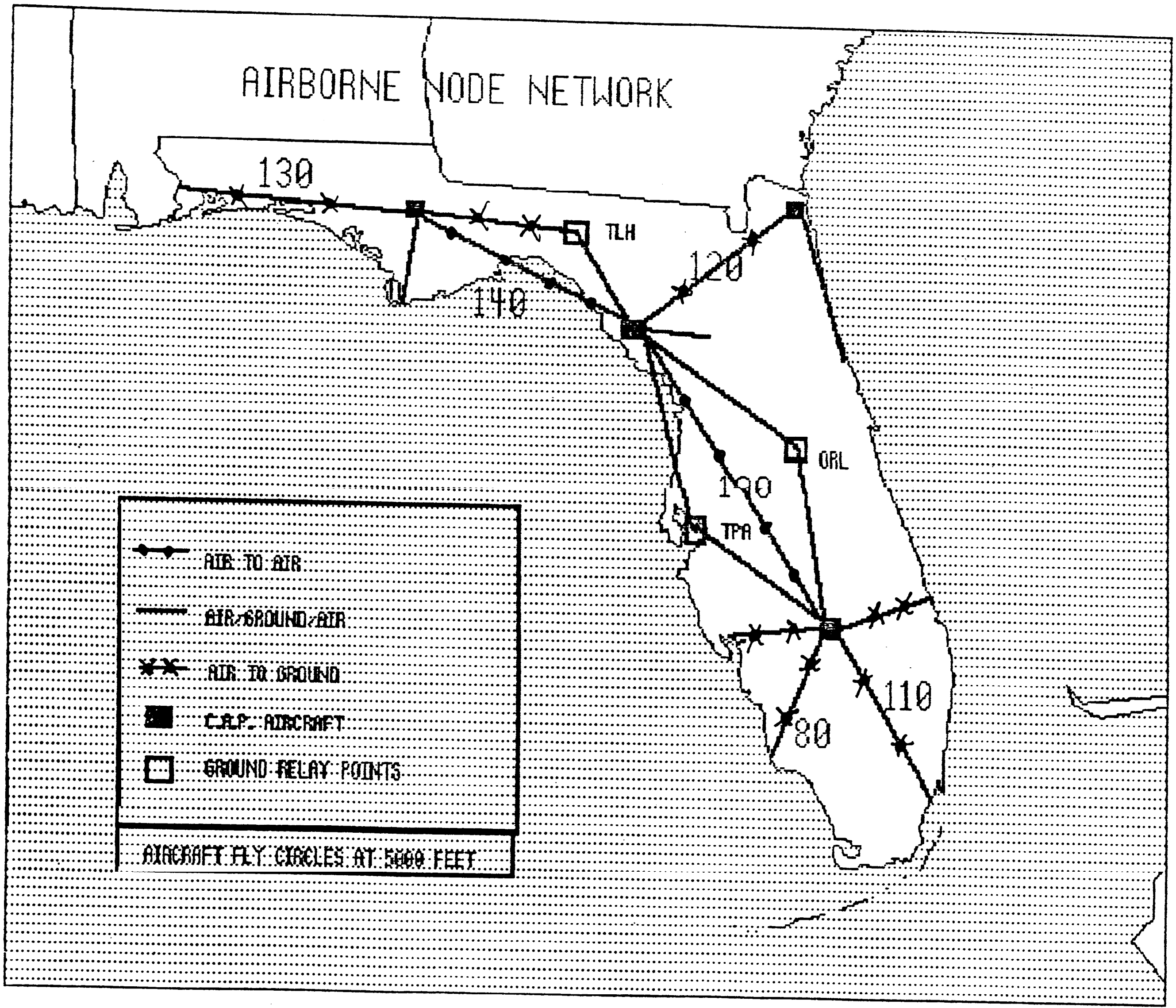


FIGURE 1: Airborne network and distances within Florida

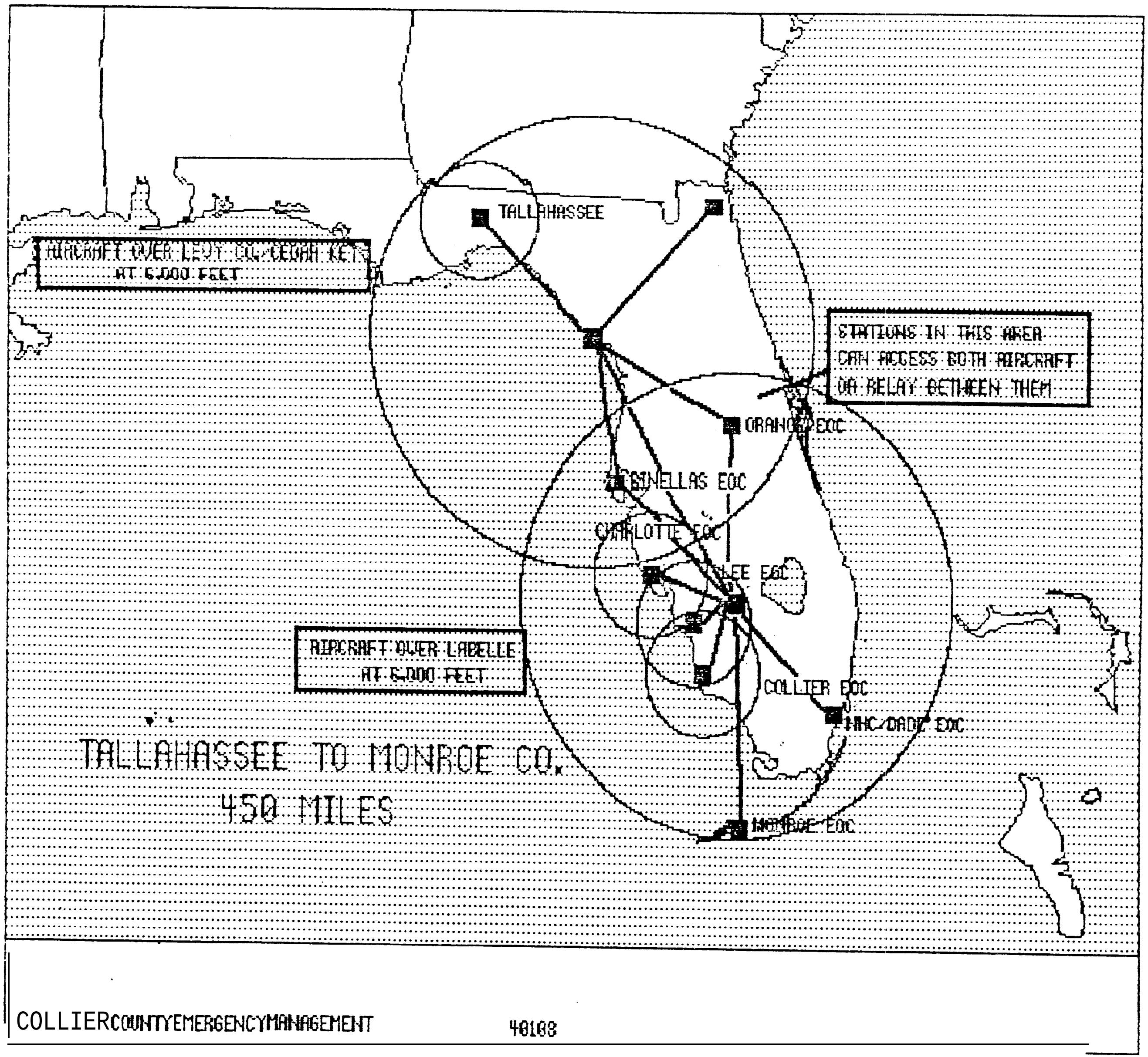


Figure 2: Two aircraft link from South Florida to Tallahassee showing approximate coverage from aircraft (large circles) and through terrestrial links normally used (small circles).

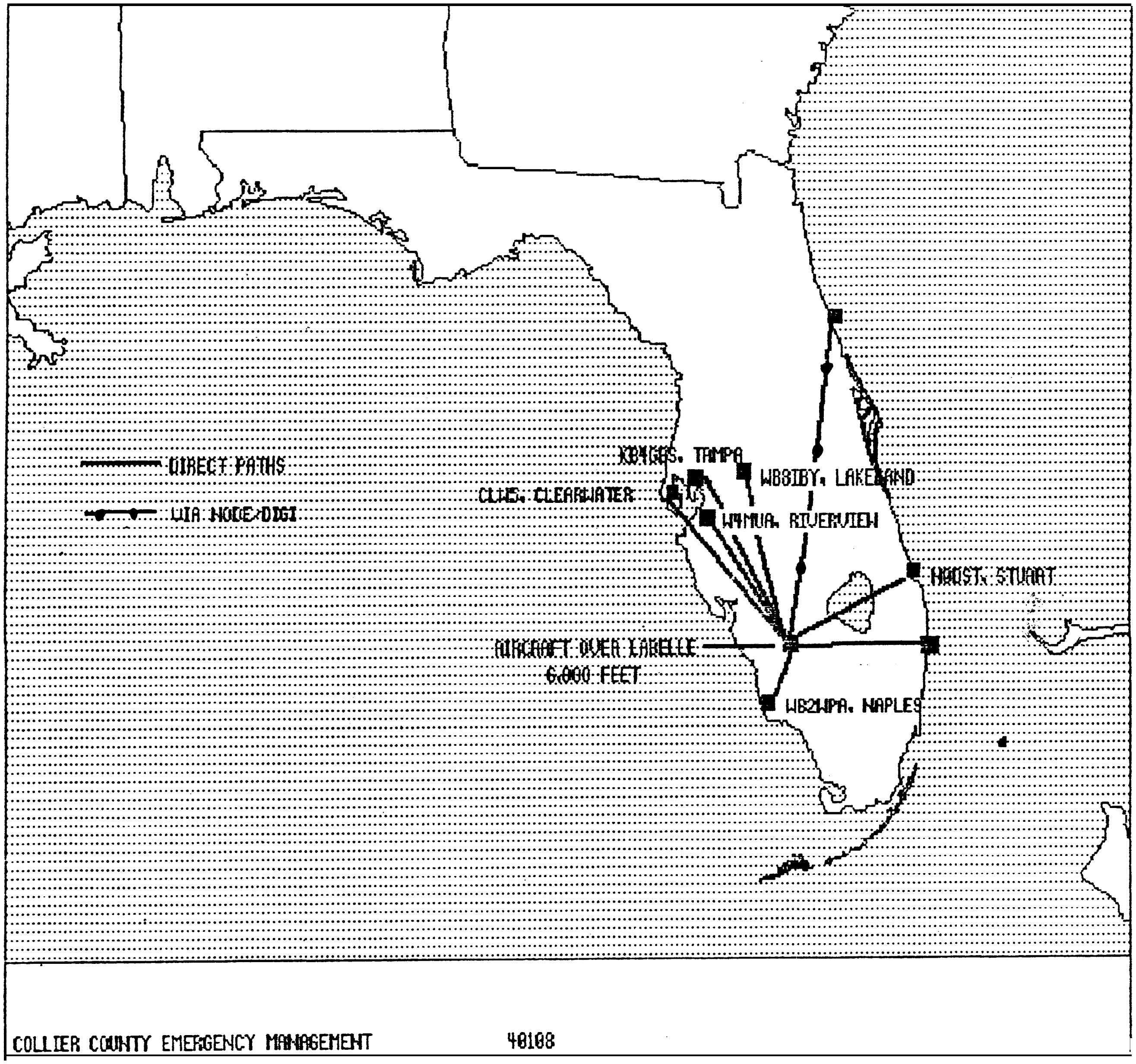


Figure 3: Areas worked via single aircraft located at 6,000 above LaBelle, Hendry County, Florida on 145.050 mHz

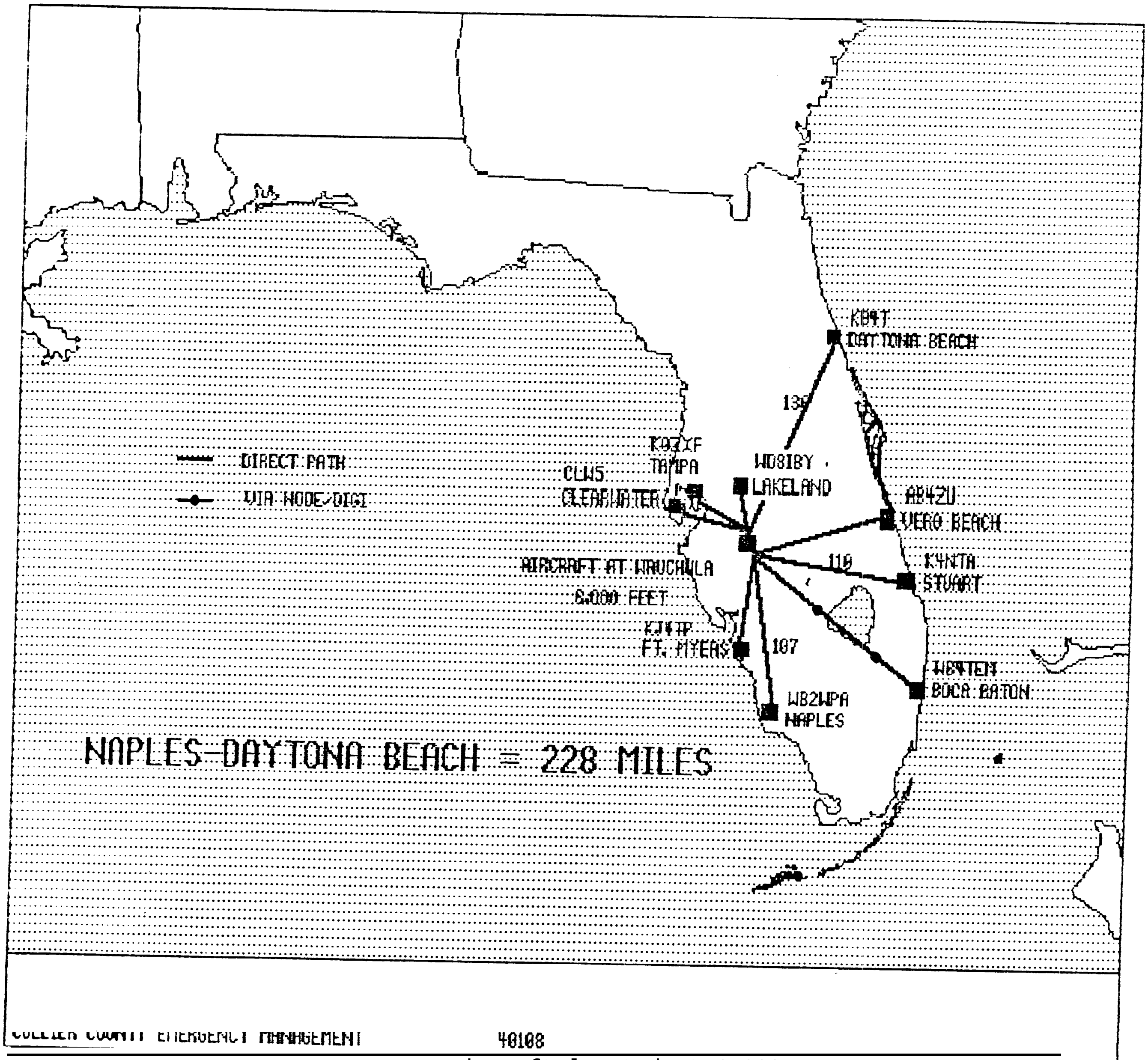


Figure 4: Areas worked via single aircraft located at 6,000 feet above Wauchula, Hardee County, Florida on 145.050 mHz, including 220+ mile path from Naples, Collier County, to Daytona Beach, Volusia County.

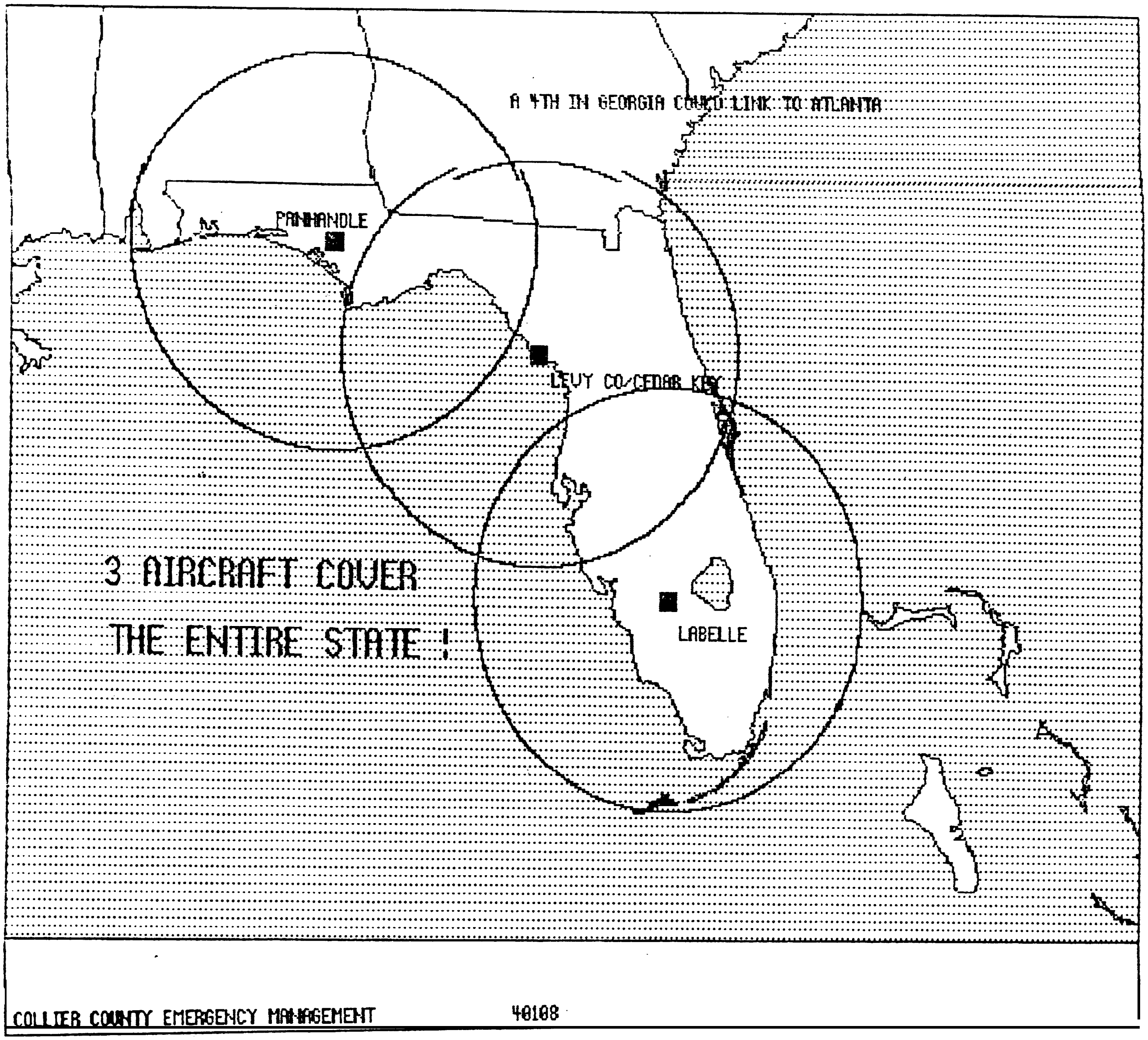

A 4TH IN GEORGIA COULD LINK TO ATLANTA
PANHANDLE
LEVY CO/CEDAR KEY
3 AIRCRAFT COVER
THE ENTIRE STATE !
LABELLE
COLLIER COUNTY EMERGENCY MANAGEMENT
40108

Figure 5: Theoretical three aircraft system, providing full coverage of the state of Florida, plus parts of southern Georgia and southern Alabama, with aircraft at 6,000 feet.

A PROTOTYPE TNC-3 DESIGN APPROACH

By Bill Beech, NJ7P
P.O. Box 38
Sierra Vista, AZ 85636-0038

Doug Nielsen, N7LEM
PO Box 1006
Ft. Huachuca, AZ. 85613

Jack Taylor, N7OO
RR-3 Box 1690
Sierra Vista, AZ 85635

Abstract

Many TNC design proposals have been advanced over the years. However, with few exceptions, the venerable TNC-2 and it's clones still proliferate the marketplace. Factors influencing this situation include: (1) A very cost-competitive market between various entry level models. (2) Proprietary operating software that has tended to stifle TNC development. This paper describes a design philosophy that the authors believe will lead to the next generation TNC.

1. Background

The familiar Z-80 based TNC was entirely adequate at the beginning of the packet revolution. The clock speed was such that it performed nicely on the fledgling network systems of the era. During this time, there was limited traffic to burden the system as well as a limited number of packet operators. The initial network configuration was composed primarily of multi-hop digipeaters located at strategic sites. The development of node software capable of being run in a low cost TNC-2 sparked a flurry of layer 314 network development that continues to this day. As BBS software's and procedures matured, more and more traffic was being moved across the fledgling packet networks. This was a self-expanding process that attracted more users to the mode...and in turn, created more system traffic. Some felt that as the market for commercial TNCs expanded, this would be an incentive for manufacturers to market models with more powerful capability. While some effort has been made in this area, consumers have tended to stick with the more economical traditional models. Noting this trend, manufacturers have primarily concentrated their efforts on software upgrades for existing hardware lines.

As the load on amateur packet networks increases, there is a greater need for a cost-effective TNC that has sufficient speed and memory capacity to handle the increased user applications. In the current market one can find a variety of TNC approaches. There is the TNC on a card that interfaces with a slot on the PC. This has worked nicely with applications requiring constant PC operation, such as BBSes and PC based nodes running G8BPQ software. There are several software approaches where an appropriate modem is coupled to a computer. Advantages with this concept include a lower unit cost and rather sophisticated operating features. Drawbacks include the need to purchase a computer if one isn't already in hand, and the possibility of requiring continuous power-up on the computer under certain circumstances.

There has been considerable interest within technical circles to develop a TNC around a digital signal processing (DSP) modem, but thus far efforts to produce an economical unit has not succeeded.

In addition to a TNC functioning as an end user device, with the proper software, it can also function as a layer 3/4 node/packet switch. In this regard the current production of TNC-2 clones have limitations. First, at speeds of 9600 baud and above, the slower processor configuration causes them to become speed-limited in handling packets on active backbone trunks. Second, an economic penalty is paid due to each radio port on a multiport network node requiring a separate TNC. At an average cost of $125.00 per TNC, a four port node would require an expenditure of $500.00. One firm currently offers a dual-port TNC, but it is priced such that a four port node would cost $600.00. However, this TNC does support data rates up to 19.2Kbaud with optional modems, which the TNC-2 clones will not do.

2. TNC-3 Design Concept

Pending introduction of economical DSP modem/TNC technology, there appears to be a definite niche in today's market for a cost effective TNC that can satisfy the needs of both end-users and node/packet switch applications. With this in mind, the basic TNC-3 is designed to handle modem speeds beyond 56 Kbaud. Since its configuration will vary with application, it can be optionally expanded. The basic unit consists of a mainboard with dual port capability and a RS-232 interface. It has a bus arrangement supporting up to two optional high-speed DMA ports and two optional low-speed ports. The application will determine the modem requirement.

For the average end user, the TNC-3 has two modems, one for 300/1200 baud and the other for 1200 baud. The 300/1200 baud modem is optimized for HF operation on the bands below, as well as at, 10 meters. Both of these modems use the time-proven EXAR 2206/2211 chipset and have true DCD. Since the average entry level HF radio has stability sufficient for packet operation, it is not necessary for the HF modem to have a tuning indicator. A sensitivity control to compensate for varying IF bandwidths for optimum packet reception is provided, Regardless of the modem type, serious HF packet operation requires a 500 Hz CW filter be installed in the radio. While Clover and Pactor are superior digital modes in the presence of QRM and fading HF conditions, they are not competitive to packets ability to support multiple users on a channel. On a clear channel with a usable signal to noise ratio, packet will have greater throughput compared to either Clover or Pactor. Additionally, contemplated revisions to the AX.25 LAPA (Link Access Protocol -- Amateur, currently in review) will improve the transmission of HF packets. It is for these reasons the TNC-3 continues to support 300 baud operation.

Software support for the TNC-3 is derived from the new proposed AX.25 LAPA protocol and it is intended the source code, which is written in the C programming language, will be publicly available for non-commercial use.

This code development is primarily due to the efforts of Doug Nielsen, N7LEM, one of the authors of this paper. This feature alone, will hasten the development of many new innovative public domain TNC programming features by those who enjoy working with code. While the basic TNC-3 is configured for 256K of RAM, it can be expanded to 1 meg. The user tradeoff here is operational need versus cost of the added RAM.

Optional plug-in modems are/will be available for 9.6, 19.2, 38.4 and 56 Kbaud. These are based on the familiar TAPR design. The 19.2, 38.4 and 56 Kbaud capacity is intended to be used on the 100 KHz channel assignments authorized for the UHF band. A TNC-3 four port network switch could support two 56 Kbaud backbone trunk channels. Perhaps one on the new 219 MHz frequency (assuming ARRL actions will be successful in obtaining it for digital use), and one on the 430 MHz band. A third port could be operating at 9600 baud as a server port on 223 MHz and the fourth port, a 2 meter 1200 baud user LAN channel.

3. TNC-3 Hardware

The TNC-3 project is an on-going effort and Bill Beech, NJ7P, has had a prototype undergoing test and evaluation since March 1993. In addition to being a platform for checking out a revised version of the AX.25 LAPA protocol, N7LEM has done a nice job of writing backwards compatible software for it. Currently a second prototype is undergoing evaluation. This prototype configuration is shown in figure 1.

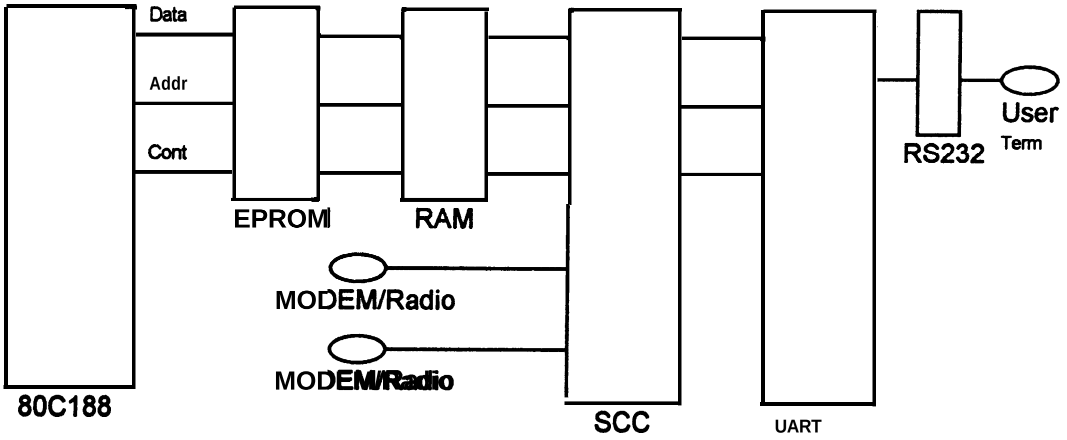


Figure 1   TNC-3 Block Diagram

The heart of this unit is an Intel 80C188 microprocessor capable of supporting up to 1 megabyte of RAM and 128 kilobytes of EPROM. The two HDLC interfaces for packet are provided by a NEC 72001 Serial Communications Controller. The asynchronous interface to the user is provided by a 8250 UART. One of the HDLC ports utilizes the two channels of DMA circuitry in the 80C188 to allow high speed operation.

The processor was chosen to be programming compatible with the IBM-PC clones. The Intel microprocessor was chosen over the NEC V40 microprocessor because it provides integrated chip select circuits in place of the asynchronous port. Since the V40 asynchronous port has no handshake pins, it was not usable for user interface. The chip select circuits reduce the need for external glue logic.

The NEC 72001 was chosen over the Zilog 85C30 because it requires less glue logic to interface to the microprocessor bus. It also provides the SCC functions without the behavior problems found in the 8530 family.

The TNC-3 was designed to accept additional I/O cards for additional radio ports and telemetry functions. These cards will provide another pair of DMA high speed ports and a pair of low speed ports. This design flexibility allows an ultimate node capacity to as much as three 56 Kbaud and three 19.2 Kbaud ports!

4. TNC-3 Software

All software development for the TNC-3 was done in the C language, with the exception of 50 lines of startup code written in assembler. The Borland runtime library was modified to run on the hardware provided by the TNC-3.

The TNC-3 currently is running with a ROM-based monitor program. The monitor, which is written entirely in C, allows the user to read and/or modify both memory and I/O ports. It also allows the loading of code, compiled on the PC, into memory. This load image can be either the DOS EXE or COM format.

The TNC-3 code was initially developed on a PC using a DRSI card. The code was then ported to the TNC-3. The code implements and was used to validate version 2.2 of AX.25 LAPA.

5. Conclusions

This design is a departure from the high speed TNC replacements normally encountered because it is developed as an integrated hardware/software solution. The designers were able to develop both the hardware and the software as one project. This concurrent development allowed optimizing the functionality while reducing the cost of the hardware. The basic dual port TNC-3 will almost be cost comparable with the TNC-2. The multiport switch will cost less than an equivalent number of TNC-2s.

DSP Implementation of 300- and 1200-Baud FSK Packet Demodulators

Jon Bloom, KE3Z

American Radio Relay League
225 Main Street
Newington, CT 06111
email: jbloom@arrl.org

ABSTRACT

This paper gives the theory of operation for a DSP implementation of **300-** and **1200-baud** FSK modems. Specific algorithms are described and filter characteristics are given for an actual implementation.

The use of digital signal processing (DSP) for packet modem implementation is clearly the technique of choice for speeds up to at least 9600 bauds. The declining cost of DSP chips makes DSP modem subsystems cost competitive with analog designs, and the increasing capabilities of these devices make design of packet modems straightforward at moderate speeds.

Packet modem standards have increased over the years, but the two earliest standards, **300-** and **1200-**baud FSK, have survived and continue in widespread use. The **1200-baud** standard, which is widely used with voice-grade VHF-FM radios, calls for tones at 1200 and 2200 Hz. (Note that packet sent using these standards is **NRZ-I** encoded, so the specification of one frequency as mark and the other as space is not necessary.) **300-baud** packet, used primarily with HF SSB radios, tends not to use a consistent pair of tones, since radio tuning can be used to adjust the received-signal center frequency as needed. A tone spacing of 200 Hz is used at 300 bauds. Implementing a phase-continuous FSK modulator for these standards is trivial, most easily done with a sine look-up table. Demodulation presents more substantial challenges, which this paper addresses.

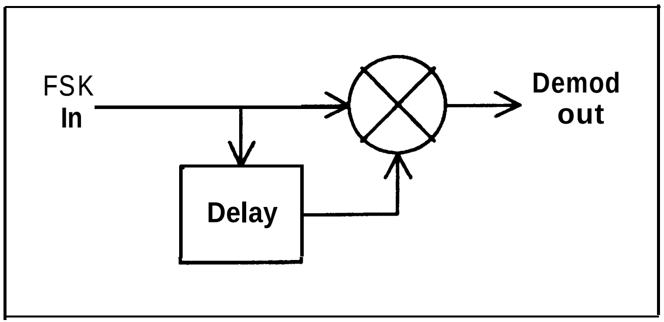


Fig l-FSK Demodulator

Basic Demodulator

Demodulation is performed using the technique shown in Fig 1. The input signal sequence is delayed by a period equal to one quarter of the cycle (90”) of a signal at the center frequency of the signal being demodulated. For **1200-baud** FSK, this frequency is 1700 Hz, and this frequency is also used at 300 bauds for convenience. The delayed sequence is multiplied with the undelayed sequence. For a **sine-**wave input, the result is as follows:

$$\begin{aligned}&A\sin(2\pi fnT)\cdot A\sin(2\pi fnT+d)=\\&\frac{A^2}{2}[\cos(2\pi fd)-\cos(2\pi fd)\cos(2\bullet 2\pi fnT)\\&+\sin(2\pi fd)\sin(2\cdot 2\pi fnT)]\end{aligned}$$

(Eq 1)

where:

- ***A*** is the peak input amplitude
- ***f*** is the input signal frequency
- ***d*** is the delay time
- ***T*** is the sample period
- ***n*** is the sample index

Note that this results in a constant term and two terms at twice the input frequency. These latter terms are filtered out, leaving the constant term:

$\frac{A^2}{2}\cos(2\pi\, fd)$. For the case when d is equal to 90° at the input frequency ($d = 1/4f$), the constant term becomes zero. At frequencies higher than that, the result is a negative level, while at lower frequencies the result is positive. Thus the multiplication of the delayed and nondelayed sequence results in a constant term that describes the frequency shift of the input signal.

A Practical Demodulator

Examining the above equation shows that a fixed delay of one $1/4f_c$ is required, where f_c is the signal center frequency. The easiest way to provide this delay is to select a sample frequency at which an integral number of sample periods equals the required delay. The first such frequency for the 1700-Hz center frequency of the modems described here is 6800 Hz. At that sample frequency, a one-sample delay corresponds to a delay of 90” at 1700-Hz. 6800 Hz is the sample rate used in this modem. Note that this allows input signals up to 3400-Hz satisfying the Nyquist criterion. Systems in which the designer is constrained to use a sampling rate other than one that provides this simple integral-multiple approach will require a filter that provides a suitable fixed delay. Such a filter is trivially implemented as a one-tap FIR filter.[l]

Fig 2 gives a block diagram of the complete demodulator. Since the output level of the demodulator described above depends on the input amplitude, an AGC stage first normalizes the input amplitude. An AGC level is computed and used to calculate a gain factor by which the input signal is multiplied. The input signal is first peak detected. That is, the amplitude of each input sample is compared to the current peak value. If the input value is greater, the current peak value is made equal to the input. If not, the peak value is multiplied by a value slightly less than 1 to decay it. This is shown in the flow chart of Fig 3. In the flow chart, x(n) is the input! sample, y(n) is the output sample, and K is a constant chosen to give the desired output amplitude.

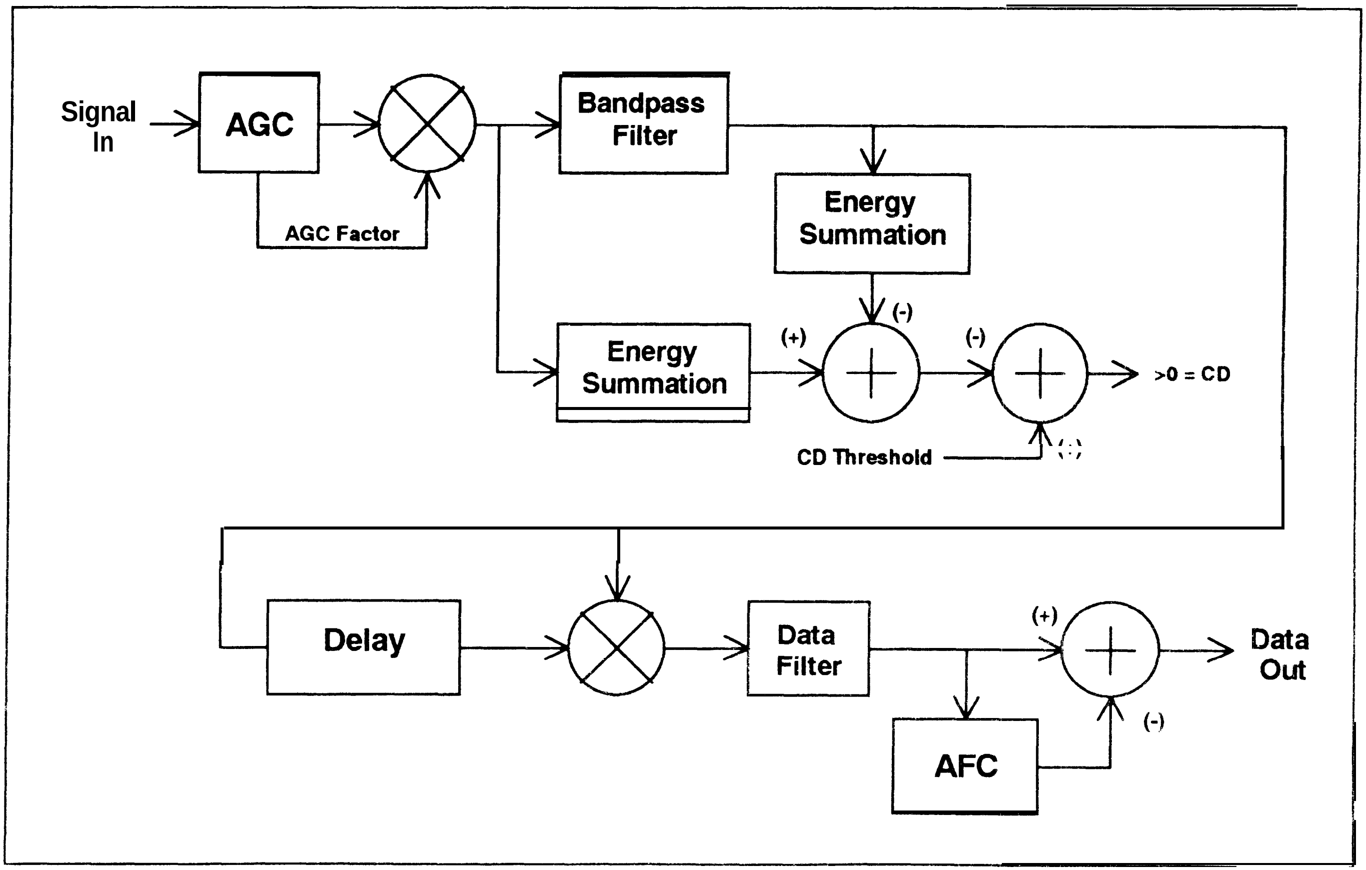


Fig 2-Demodulator Signal Flow Diagram

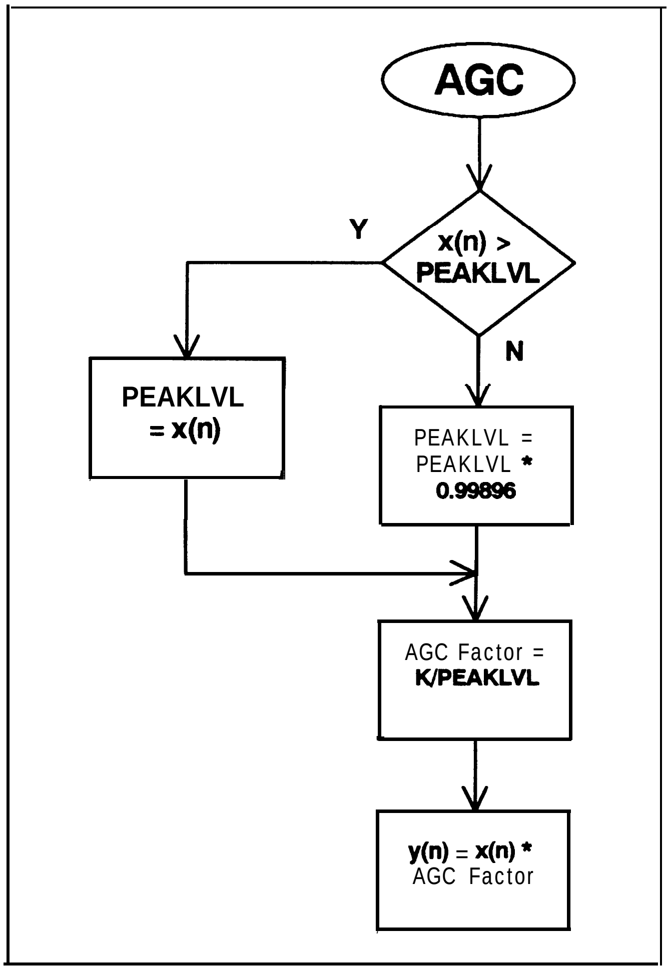


Fig 3-Flow chart of the AGC computation.

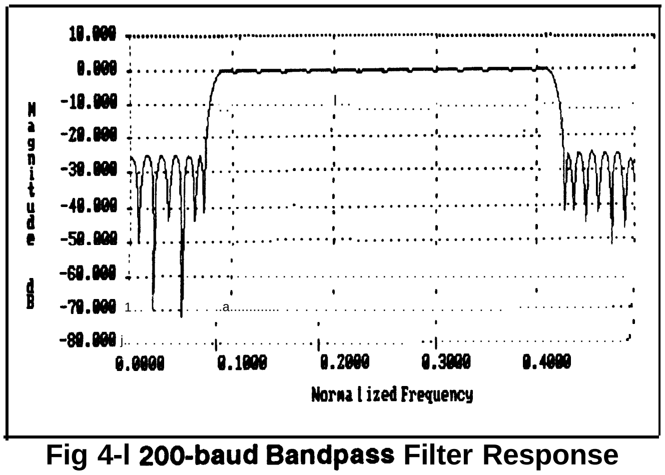


Fig 4-l 200-baud Bandpass Filter Response

Carrier detection (CD) is done by comparing the energy in the input signal to the energy in a bandpass-filtered copy of the signal. When the input signal is composed of noise, much of the energy of the signal will fall outside the **passband** of the filter. Thus there will be a substantial difference in the amount of energy in the two signals. But when the input signal is the desired FSK signal, the filtered and unfiltered signal will be nearly the same. Calculating the difference between the two signal energies and comparing the result to a fixed threshold develops a carrier-detect decision. The amplitude response of the **1200-baud bandpass** filter is shown in Fig 4, the **300-baud bandpass** filter in Fig 5.

Calculation of the energy of the signals is performed over blocks of 512 samples:

$$E = \frac{1}{512}\sum_{i=0}^{511}[x(i)]^2$$

At 6800 samples per second, this results in an updated CD result about every 75 milliseconds. Using 512 samples minimizes errors caused by having an aperiodic signal represented by the samples.

The bandpass-limited signal is applied to a detector like that described above. Band-limiting the signal ensures that no (unwanted) low-frequency signal present in the input audio produces a $\sin(2\cdot 2\pi fnT)$ signal (see Eq 1) that is in the **passband** of the data filter.

The detected signal is filtered by a low-pass filter that rejects signals above 600 Hz **(1200-baud** demodulator) or 150 Hz **(300-baud** demodulator). This filter removes the unwanted signal components from the detection process and removes any **high-**frequency noise on the detected signal.

An AFC process is used on the detected signal. If the FSK signal center is 1700 Hz, the positive and negative peaks of the detected waveform will be equal in amplitude (but negative in sign). If the signal center is not 1700 Hz, these peaks will be of different amplitudes. Note that this is ***not*** the same as saying there is a DC offset in the waveform. There may **be**—almost certainly there ***will*** be-a DC offset imposed by the data stream. Thus the AFC has to operate on the peak values of the detected signal rather than the average value. Attack and decay time constants are used to minimize AFC jitter. The AFC process is shown in the flickered of Fig 6. Note that it depends on there being continual data transitions, as are ensured by the use of HDLC packet framing. This technique would not be useful for an FSK system that did not ensure data transitions (e.g., asynchronous **RTTY).**

The AFC function can correct small tuning errors, although large offsets will cause part of the

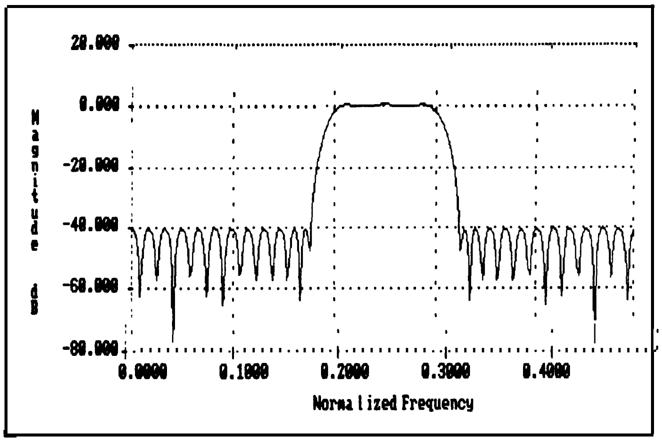


Fig 5—300-baud Bandpass Filter Response

signal spectrum to fall outside of the bandpass filter. AFC also helps during HF selective fading, as the reduction in amplitude of the fading tone causes the AFC to skew the decision point on the detected waveform. In the worst case, the complete absence of one of the tones, the detector becomes an AM detector, with the decision point being at one half the peak detector output.

Conclusion

While the modem described here has not been subjected to detailed testing, on-the-air experience and side-by-side comparison with several other packet modems has shown it to perform well. The author acknowledges that 300-baud HF packet and 1200-baud VHF packet are not the best possible schemes. Still, DSP-based packet modems will likely need to include these modes to maintain compatibility with older, nonprogrammable packet modems. It is the author's hope that those who are implementing new DSP packet modem systems will be able to use the information contained in this paper to implement compatible modems with acceptable performance quickly and easily, the better to concentrate on development of newer, better modulation schemes.

Notes

[1] Phil Evans, "Implementation of an FSK Modem Using the TMS320C17," ***Digital Signal Processing Applications with the TMS320 Family,*** ed. Panos Papamichalis (Englewood Cliffs: Prentice Hall, 1990) 363-365.

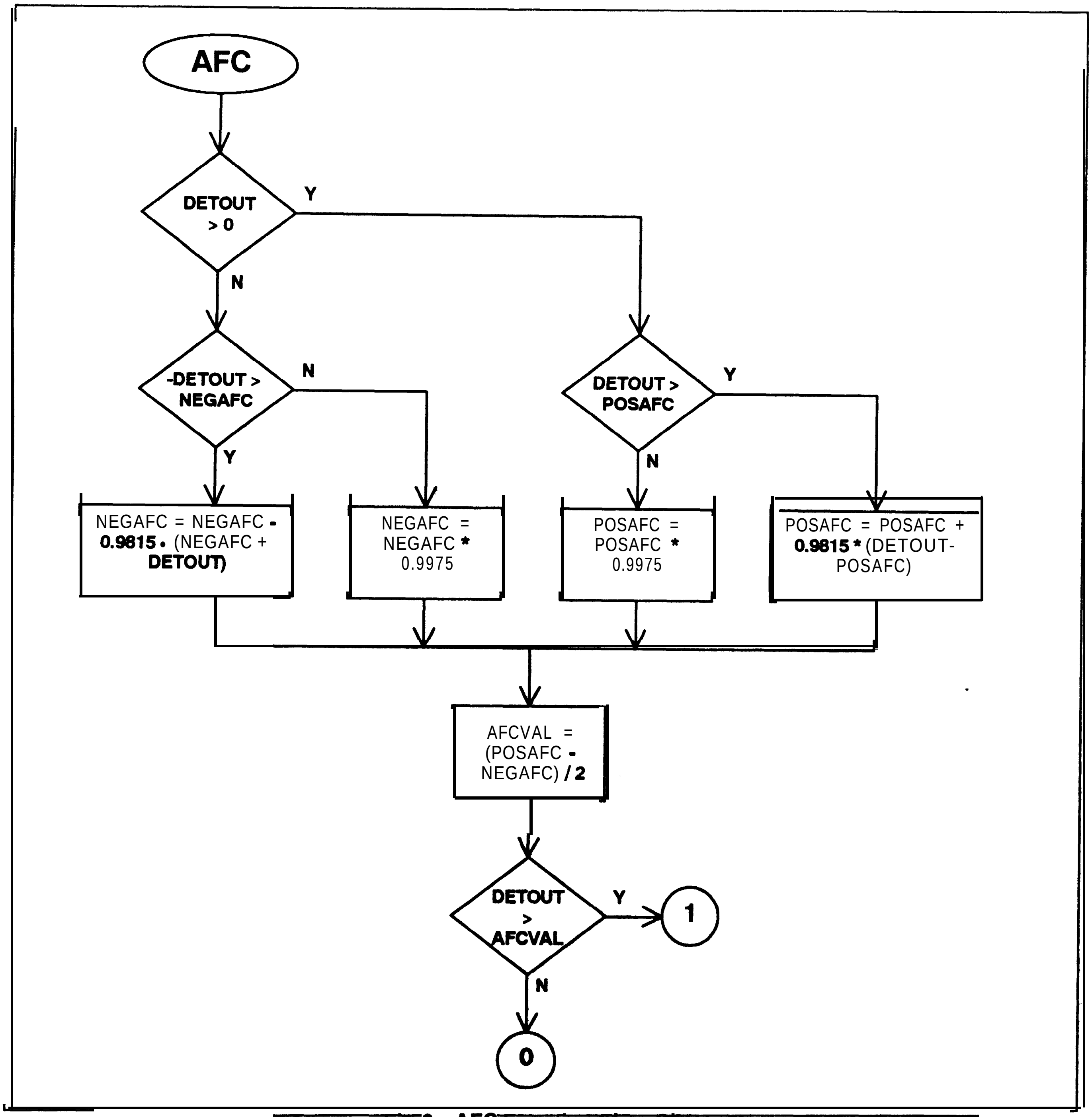


Fig 6—AFC Function Flow Chart

The ARDS PROJECT

By **WØLIQ and K9LTL** -- 7/93

INTRODUCTION--ARDS is an acronym for Amateur Radio Data Syntheses. Three kinds of systems can be synthesized: manual, computer-aided, and automated. We have only tested models of the first two via using simulation. Meanwhile, we're awaiting permission from the F.C.C. to try over-the-air tests. Part 97 amateur radio rules forbid secrecy. A secrecy waiver is necessary because **ARDS'** radio signals cannot be deciphered without ARDS systems. This paper **reveals** how **computer–** aided systems work and how they can supplement ham communications.

ARDS PROJECT **INITIATION**--Creativity has been defined as the ability to see CONNECTIONS between things. Inspiration for ARDS **came** from an account on how the Rosetta stone was discovered and interpreted by French scholars to decipher Eygptian hieroglyphics. (The account can be researched easily in libraries). Decipherment became possible by using **Demotic,** an intermediate language, to find connections between hieroglyphics and Greek. The Stone had been inscribed during 196 BC. We wondered, why couldn't one develop a computer-assisted language **to translate** English into Latin-based languages and vice-versa? The increasing availability of personal computers made it an attractive possibility to explore. We know hams have more connections between languages and cultures than the general population. ARDS project was based on those insights in 1989. The project has progressed since then beyond our expectations. Research has been aided by support from several other hams who volunteered to help us.

Meanwhile, we've read reports **about commercial translational** aids which employ "artificial, intermediate" languages. However, we don't anticipate they will be practical for ham **useage.** Also we have analyzed several translational aids developed BY hams FOR hams. We are unaware of any publications of this sort which were designed to utilize digital processing technologies.

ARDS' COMPUTER-AIDED SYSTEM STATE--Our Model **12** has the following system components: 1. Datascript digital alphabet; 2. **Hamtalk** language with tutorial plus nomenclature; 3. Printed AND software **files** which list indexed text data; 4. CHAT and DATA modes for operations; and, 5. P/C computer hardware with DOS 3.3 or higher **O/S. (Item 5.** is user-provided because ARDS is computer-hardware independent).

ARDS' COMPUTER-AIDED SYSTEM PROCESS--Model 12 enables hams to communicate conventionally in the CHAT mode to initiate contacts. DATA mode is entered to exchange remote control signals. Those consist of Morse code or English words. Automated systems have potential for development (assumes international **ARDS'** standards could be set).

FUNDAMENTAL SYSTEM CONCEPTS---If hams **obtained** data files which will correspond with each other's data records, then that data could be referenced **via** remote-control. In other words, only DATA INDEXES or (DIs) must be exchanged. This **OLD IDEA** has been **demonstrated** in part **by Q-codes,** ARL numbered messages, etc. Importantly , personal computers make **it quite** practical to store **text** and other data in forms which can be manipulated. Text can be be **joined** together quickly and displayed on monitors, on printers, or spoken over audio speakers.

In conventional communications (conducted in CHAT mode), cw or voice signals are exchanged letter by letter or word by word. But in DATA mode, DATA INDEXES are exchanged to access, retrieve, and manipulate pre-stored data. This data synthesizing process displays messages in upper and lower case letters on monitors and/or print-outs. With optional software, text data can be heard as speech.

These data need not be limited to text messages (assuming- F.C.C. approval). Multimedia data could also be synthesized in an analogous manner. Four character control signals can generate lengthy messages almost instantly and in real time. “Hamtalk” language is composed of " Datascript” alphanumeric strings. By selection, they are the SAME characters you’ll find positioned on keypads of 101 style computer keyboards (U.S., South American, and Western European).

The electronic age was born when Lee De Forest announced that he could control electron flow in diode tubes with grids. De Forest enabled engineers to incorporate GAIN into circuitry. Analogously, microprocessor inventions and programming techniques now enable system designers to exploit “DATA GAINS and DATA ACCELERATIONS”. These new exciting phenomena may someday be appreciated as REVOLUTIONARY. Present and future digital technologies make the phenomena possible.

DATA FILE ORGANIZATION--Model 12 “demo” file includes Q-codes, QN–codes, ARL numbered messages, alphanumerics, and other topics which have popular utility for ham activities. Brief numeric indexes were assigned to full sentences, partial sentences, questions, words, and alphanumerics for convenient referencing. They’re delimited by using mnemonic punctuations (periods terminate data; hyphens bring data together). Text data records hold up to 40 characters. Records will be printed consecutively if more characters are needed. This feature is critically important for text overflow from translations. A TOPIC MENU helps users find data via topics and data indexes via ranges.

EXAMPLES OF COMPUTER-AIDED OPERATIONS--Initial contacts will be made conventionally in CHAT mode by sending protocols analogous to those in practice now. File sync codes must be announced during CQs. Upon making contact, operators will begin and end transmissions in CHAT mode, Also, they may TOGGLE into DATA mode by exchanging signals for the plus key (+ is a character in Datascript alphabet). For example, CW or SSB signals can be copied to display VDT messages as follows:

	chat mode data mode chat mode	
/CW MORSE/	QSL DE KSLTL +121.123. + QSL?	
or		
/SSB HAMSPEAK/	QSL DE KSLTL, ADD, ONE TWO ONE, POINT, ONE TWO THREE, POINT, PLUS, QSL?	
		vdt details
[VDT DISPLAY]	QSL DE KSLTL +	[CHAT mode copy]
	Can you copy me?	[Data Index 121]
	I copy you solid!	[" " 123]
	QSL?	[CHAT mode copy]

DETAILS: Hamtalk digital language can be exchanged as Morse CW or in the form of “Hamspeak”, a voice option for SSB useage. The VDT display will be the same if either signals will be inputted. Plus symbol is signaled by “ADD or PLUS” to toggle DATA mode "On" or ‘Off”.

NUMBERS SYNTHESIZED ON-THE-FLY (Use of RAM Buffer)

Data records for numbers (1, 2, 3, etc.) were assigned the same data indexes. Therefore, numbers can be sent as is, OR, increased in value by synthesizing them on-the-fly with hyphens as follows:

```
                      chat mode   data mode         chat mode
   /CW MORSE/         R R DE K9LTL +203.207-5-7-9-. + QSL?
                                                   vdt details
   [VDT DISPLAY]      R R DE K9LTL +                [CHAT mode copy]
                      I'll send signal report later. [DI 203]
                      Your report is QSA/RST_579   [DIs 207 etc.]
                      QSL?                          [CHAT mode copy]
```

DETAILS: On 101 keyboards, the QWERTY sides function as electronic typewriters in CHAT mode. Hamtalk signals are copied on keypads in DATA mode. Minus and decimal keys can be used for hyphen and period keys. Calculate your character gains by comparing CW characters with those actually displayed on VDTs. Text “accelerations” can only be appreciated by pressing decimal keys. Period "print commands" cause messages to explode on VDTs. The underline after "RST" is a spelling device which provides a space and prompts “data must follow". Also, observe that displays are printed in upper and lower case letters. A CHAT/DATA mode prompt keeps operators informed of mode status on the bottom line of monitor screens.

ARDS AS A TRANSLATIONAL AID--If files are translated into foreign languages, they can be understood by foreign hams...and vice-versa. This principle was tested via simulation with XEICFO in Mexico. A cassette tape was furnished with Hamtalk Morse signals. Carlos was able to interpret Hamtalk by referencing a printed file written in Spanish. He completed a questionaire which verified that he fully deciphered Hamtalk language. Here’s an example of how English and Spanish computers display the same CW Morse signals in both modes:

```
                        chat          data  chat
   /CW MORSE/         XEICFO DE K9LTL +733. + ?

   [ENGLISH VDT]      XEICFO DE K9LTL +                          [CHAT mode]
                      QSB-Signal is fading.                         [DI 733]
                      ?                                          [CHAT mode]

   [SPANISH VDT]      XEICFO DE K9LTL +                          [CHAT mode]
                      QSB-La señal está desaparenciendo.            [DI 733]
                      ?                                          [CHAT mode]
```

DETAILS: The Spanish translation happens to be longer than than the message in English. If longer than 40 characters, a second record would have been displayed on the next line to complete the message.

CONCLUSIONS: We have proven to our satisfaction that ARDS systems have potential as communicative tools. Important questions remain. Can we reach an agreement on ARDS standards? Will there be data censorship arising from governmental regulations ? Once questions are resolved, we expect ARDS developments will continue. For more ARDS' details, send questions and a S.A.S.E as follows: Roy Ekberg, WØLIQ, 2111 E. Santa Fe, S 346, Olathe KS 66062. (Tel: 913+242-8287)

USAGE OF TCP/IP
OVER ROSE IN THE TAMPA BAY AREA

Chuck Hast, KP4DJT/TI3DJT
Tampa, Florida

With the usage of NOS in the Tampa Bay area, we have found the need to link between NOS users using the ROSE network. Like other areas where ROSE is used as the primary networking system, up until a year ago, the only way to do so was using SLIP over TNC's running in transparent mode. Manually setting up the link then issuing the slip connection to the distant IP facility.

About a year ago Tom Moulton told us he was ready with code that would pass non $FO PID's specifically to allow IP to pass over the ROSE network. Of course, NetRom or any other non $FO PID could be passed thereby allowing the identification of say packet voice, images, or whatever needed.

As soon as we had the new code in our switches I started testing with other stations in the area to determine if what was promised would work. At first we found that the "JUMP-START" command and the ROSE link status messages would cause the NOS code to route the link being set up to the AX.25 BBS in NOS. We were able to circumvent this problem by turning off JUMP-START and using the "reliable" mode of ROSE, which does not generate the status messages.

After setting up the initial links, we tested with KO4KS, N4ZXI and the TBARS BBS, and found that we were able to reliably exercise the different facilities in NOS, starting with the humble "PING" session and terminating with FTP and SMTP operations.

A result of this testing was Brian, KO4KS including in his version of NOS (TNOS) a separate JP call, this call would take any non IP PID and disregard it, but it would respond to any links arriving with a IP PID, this action took care of the $F0 PID problem, since TNOS has a separate call for AX.25 connects.

Brian has worked to make TNOS more ROSE compliant, and has not had to do any major work there, the big advantage being that ROSE does not require a special driver like NetRom does, you do not have to "fake out" the ROSE switch with a PID it recognizes, you just pass on the IP PID. Brian tells me that this alone means that NOS could be reduced by about 40k in size, if NetRom did not have to be supported in this way.

I also set up a separate NOS system on a laptop so I could observe both sides of the link, mainly to determine what needed to be done on both the originating station and the terminating station.

This link entered the ROSE network at the Tampa switch and exited at the Clearwater switch. (I can access both switches from my location.) The path took the data through 4 switches and over a very congested part of the old 220mhz flat network (which we are trying to abandon as soon as time and money permit).

I found that a FTP would run quite well, I was able to take advantage of Phil Karn's implementation of the "M" bit to send large IP frames which were fragmented at the AX.25 level, sent over the network to the distant NOS system where the 1 k IP frame was reconstructed.

Sending a IP frame of 1 k in size resulted in five 256 character frames at the AX.25 level, the first of which had the IP header and part of the data, the rest were data. The first 4 frames were sent with the 'M' Bit set and when the final frame was sent the 'M' bit was flipped allowing the NOS system on the far end to reconstruct the 1 k IP frame.

The usage of the 'M' bit and AX.25 fragmentation without added overhead is a great advantage over the present system used in most areas where IP is sent over NetRom (or it's clone TheNet) with 40 characters of IP header plus 20 characters of NetRom header making a total of 60 characters for each 256 character frame, leaving only 196 characters of space available for data. Adding this gain to that of the usage of the 'M' bit makes for a very efficient transfer of data between the two NOS boxes.

The level 3 frames of ROSE add characters minimum of data to the level 3 I frames, and it is done transparently, in other words you do not have to subtract the ROSE overhead from your paclen and set accordingly, the ROSE frame is actually longer than 256 bytes by the number of level 3 bytes added to it.

As a final test, at this time I periodically set up a SMTP session with K5JB in Oklahoma City, all carried over ROSE. I believe the number of switches on that path, counting those on the Tampa, FL Dallas, TX wormhole are about 15 switches. These sessions though slow (there is a bad path on the north end through a digipeater) do work, and ROSE passes the IP PID all the way from end to end. If for some reason there is a network failure TCP picks the link back up and finishes the job.

Reviewing our findings;

IP can and will work quite well over ROSE. The combination of the two systems has added benefits for both.

Phil Karn's usage of the 'M' bit in NOS allows NOS and ROSE to work hand in hand to reduce network overhead but to still keep the advantages that each brings to packet radio.

With the implementation of the 'M' bit and PID transparency of ROSE we have a network that can reduce IP overhead and at the same time recover from a network failure, picking up where things left off.

Brian Lantz, KO4KS has implemented several servers in TNOS, one of which is the INFO server, we are now routing certain types of calls to INFO in the Tampa area to that server, it allows us to give the user much more detailed information about the area facilities, also it is set up such that it provides that information by title, so if a user is interested in only a part of the network, that user can select on that information he/she needs and not load up the network with useless data.

The conference server found in NOS has also been worked over by Brian, he has also made it such that a quick WHO will tell you not only the usual information but also the ROSE X.121 if a link is arriving over a ROSE path, this lets you rapidly identify where those users are on the network.

We have not yet been able to test the linking of two conference servers over ROSE yet, but such a link would be advantageous to all involved, the linked conference servers would reduce the number of frames going to users, at this time we have a lot of connects to the conference server from the Texas and Oklahoma area, if there was a conference server running out there it would mean that the network would only have to carry one copy of each frame between Tampa and Dallas, not one for each user.

ROSE has a much better defined addressing system for a planet wide network IP handles the end user address quite well, the combination of the two gives a much more workable addressing system, reducing the need for IP addressing for simple switching facilities, and leaving these addresses available for end users to occupy.

ROSE being based on CCITT X.121 addressing already has a global networking concept built into it, there is no need to use dwindling IP addresses for networking facilities, let ROSE take up that job and use the IP address as the address of the end facilities at each NOS or TCP user that really needs them.

NetRom, TheNet sysops note, you can use ROSE as your backbone and keep your nodes at the front end for your users. When a link is established, it is done over ROSE, this has the advantage of not needing to limit the routes and nodes to only so many deep, you can then let your users use the old system (they would not need to learn the new one since they will be connecting to other nodes and out to the target station at the other end. You would of course need to use versions of TheNet which allow connects over digipeater paths, versions such as TheNet 2.10 will not allow this type of operation.

The advantages gained by using one of these systems over a ROSE backbone are many, your nodes do not have to keep such a extensive list of nodes with your backbone being all ROSE only the exit points will be kept. If there is a network failure, ROSE will inform you of the area in which it took place and the type of failure. And if you choose to run another protocol over your backbone, ROSE with it's PID transparency will allow you to do so, something that is not able to be done with TheNet/NetRom in it's present form.

I believe the results of this work will show that the old protocol wars were fought for nothing, that like all things, you must pick and choose, and combine for a given application, here we have two systems, at one time opposites in our packet radio activity being joined, each for what it offers, both will come out the stronger for it, and the sum of the whole will be a powerful networking and user system offering very interesting possibilities for the future of Amateur Packet Radio.

There is one outstanding activity that I would like to do sometime, compare a similar NetRom link to the ones we have tested as described above. We do not have a NetRom network in this area, and I do not have the time to set up a simulation, perhaps we can do so in the future.

UNIQUE IDENTIFICATION SIGNALS FOR CCIR 625

Michael J. Huslig
Kantronics

When an AMTOR master (CCIR 476) calls an AMTOR slave, it uses a 4-character identification. Since the AMTOR alphabet is based on the RTTY code, these four characters can only be uppercase alphas [A...Z]. But an amateur call sign can be from 3 to 6 characters, of which at least one is a number. The amateur community is using some workable algorithms to map callsigns to these 4 characters; unfortunately there is no way to map these 4 characters back to a unique callsign.

When CCIR 625 was added to AMTOR, the number of identification signals expanded to 7. However, because of a checksum calculation that is used during the initial connection, the 7 characters can only be a subset of the uppercase alphabet. Because the checksum uses a modulo-20 addition, only 20 alphas can be used out of the 26:

[ABCDEFIKMOPQRSTUWZ]

Again, some algorithms are being used to convert amateur callsigns to 7 character identifications. One way is to map the 6 invalid alphas and 10 digits to the 20 valid alphas. But the mapping is not unique when trying to convert back. The algorithm is workable in that it is unlikely that two call signs which would have the same 7-character identification would be on the same frequency at the same time.

CCIR 491-1 Annex II shows an algorithm to convert a g-digit number to 7-identification characters. The identification characters can then be converted back to the same g-digit number. The algorithm basically takes the g-digit number and divides it successively by 20, mapping the remainders [O... 19] to the 20 alpha characters in the order [VXQKMPCYFSTBUEOIRZDA]. Note that the number of valid identifications (20*20*20*20*20*20*20 = 1,280,000,000) is greater than 999,999,999. This shows that there are some valid 7-character identifications that will not map to the g-digit formats.

An amateur call sign can be from 3 to 6 characters long. Let us extend all callsigns to 6 characters by padding spaces on the right such as "WØXI ". A brief examination of call signs will show that there can only be uppercase alphas and numbers in the first three positions and that there can only be uppercase alphas or spaces in the last three positions. Of course, there is no such valid call sign as “555 I ", but as you will see, there is no need to make the algorithm more complex. There is therefore, the possibility of 36 symbols [A...Z0...9] in the first 3 character positions and 27 symbols [A...Z] in the last 3 character postions. Since invalid call signs are included, the number of valid callsigns is less than 36*36*36*27*27*27 =918330048. A proper algorithm will convert an amateur call sign to a g-digit number which can be converted to a unique 7-character identification. The reverse process will take that 7-character identification and convert it back to a unique call sign.

Kantronics KAM has been using such an algorithm for some time. Consider the call sign to be a six-character string, c[0] through c[5]. For example, WØXI becomes c[0] = 'W', c[1] ='O', c[2] = 'X', c[3] ='I', c[4] = ' ', and c[5] = ' '. Each character c[i] must be converted to a number n[i]. Since the first three characters can be numbers or letters, a different conversion is used for them compared to the last three characters. For the first three characters, 'A' =0, 'B' = 1, ... 'Z' = 25, '0' = 26, '1' = 27, ... '9' = 35. For the last three characters, since they can only be letters or space, 'A' =0, 'B' = 1, ... 'Z' = 25, ' ' = 26. In a programming language, this could be expressed as:

```
if(i < 3)
        {
        if(c[i] > = 'A' && c[i] < = 'Z') n[i] = c[i]-'A';
        else n[i] = 26 + c[i]-'0';
        }
else
        {
        if(c[i] > = 'A' && c[i] < = 'Z') n[i] = c[i]-'A';
        else n[i] = 26;
        }
```

The 9 digit identity number N can now be computed using the formula:

$$N=((((n[5]*27+n[4])*27+n[3])*36+n[2])*36+n[1])*36+n[0];$$

CCIR 491-1 would convert N into the 7-character identification. At the other end of the link, CCIR 491-1 would convert the 7-character identity back to the same g-digit identity N. From N, n[i] can be obtained by successive divisions with remainders as follows.

N/36 = N1	with n [0] remainder,
N1/36 = N2	with n [1] remainder,
N2/36 = N3	with n [2] remainder,
N3/27 = N4	with n [3] remainder,
N4/27 = n[5]	with n [4] remainder.

Table lookup will get back c[i], mindful of the differences between the first three and the last three characters.

Note that not every g-digit identity can be converted to a callsign. Even if it can be converted, it may not be a valid call. Since a standard similar to this algorithm has never been specified, the 7-character identity can be anything. Every call sign can be mapped uniquely to a 7-character identity, but every 7-character identity can not be mapped uniquely to a call sign.

This algorithm does provide a unique 7-character identity for any amateur call sign. The only drawback may be that someone monitoring the channel in a 'dumb' listen AMTOR mode would not be able to do the conversion in his head; WK5M would appear as 'OCIFRDC'. However, a 'smart' listen AMTOR mode might be able to do the translation for the listener. And besides, WK5M will still respond to a call directed to 'OCIFRDC'.

Kantronics considers this algorithm to be in the public domain; anyone may use it.

Network Information Services

by Brian A. Lantz/KO4KS
TPALAN Tampa Local Area Network

INTRODUCTION

In the early days of packet radio, you were lucky to find a station or two that you could connect to for a digital QSO. As time has progressed, BBSs took the role of "generic information providers", with bulletins, and E-Mail as their strengths. Current BBSs are used nearly 100% for message handling.

The BBSs have a distinct weakness in the area of providing information to the user. The two main ways that BBSs provide generic information are:

1) Requiring the user to read certain messages

2) Requiring the user to download certain files.

Both of these provide a learning hurdle to a new user, which will probably cause the user to try to "stumble through it" without the needed information.

ROSE (and other networking tools) have taken the first steps of network information services by placing a screenful of text in the memory of the switch/node to answer a few questions the user might have. This is made simple for the user, requiring him to connect to an alias; no new actions to learn.

The average new user has limited computer experience and very probably NO experience with other E-Mail systems. Networks are a concept that scares even many long-time packeteers. An information provider should take these into account and also provide a way to be interactive, at least to a point.

TNOS NETWORK INFORMATION SERVER

One of the new, unique features of TNOS is the Information Server. This is a Hypertext driver that allows tutorials, help systems, on-line surveys, etc. to be easily added to NOS. Sub-directories can be added easily to allow multi-level nesting within each Server.

The user connects to an alias (i.e. "INFO") and is presented with a numbered menu of available options. Each entry can be either a Service file, or a Subdirectory file. Their location and all other details are not needed by the user; he simply selects what he wishes from the menus.

The whole Information Server process is driven by standard ASCII text files. Lines that begin with a tilde ('~') are control lines. All other lines are simply text. These files can be uploaded from the BBS prompt, or can be automatically uploaded using a Request Server like the TNOS REQSVR.

TNOS supports three Information Servers, which all act alike, but have different aliases, directory structures and Service files. The three standard TNOS Information Servers are INFO, NEWS, and TUTOR. INFO is used for informational content; network access frequencies, BBSs available, etc. NEWS is reserved for Amateur Radio related news. The TUTOR Server is generally used for more interactive Service files. Samples include surveys, tutorials, databases, and much more.

This rest of this paper will (hopefully) serve to wet your appetite and also to document the file format used by the TNOS Information Server until better documentation is available.

NECESSARY COMPONENTS

There are only two necessary components of a file that is used as either a Subdirectory File or a Service File for the TNOS Information Server, a File Name and a Title Line.

FILE NAME:
We all can probably agree that one of the worst "features" of MS-DOS is the size limitation of filenames; 8 character name with 3 character extension. This leads to VERY CRYPTIC filenames, at best. One definite design decision of the Information Server was to NOT inflict upon the user the responsibility of having to make sense out of MS-DOS file names. With the Information Server, you can use whatever filename you want. The file MUST have a ".tut" file extension, though.

Should you use descriptive filenames? Who cares! The user never sees thern, they see the description from the file's title line, and a number that they use to choose the Service File.

Each file must be in the proper directory for the server desired. These are (by default) "spool/INFO", "spool/NEWS", and "spool/TUTOR".

Those familiar with NOS will note that the NEWS directory is used in all other variations of NOS for the NNTP directory. TNOS names the NNTP directory "NNEWS" by default.

TITLE LINE:
The first non-blank line of the Service File is special! It serves to provide the Information server with the description of this particular item. The title line can also allow you to nest into sub-directories easily. This allows sub-menus very painlessly.

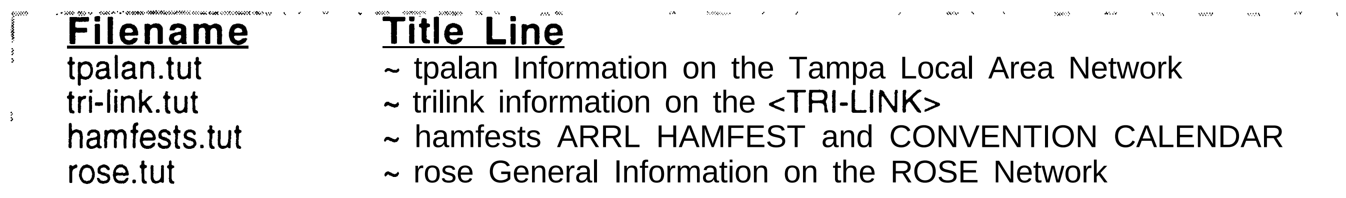
The title line can start with preceeding spaces or tabs (they are ignored). This allows you to center the line, since this is the first line displayed to the user when the Service File is executed. The rest of the title line will be displayed as the file's description in the Information Server menu.

If the title line starts with a tilde (~), it expects a line of this format:

~ subdir description

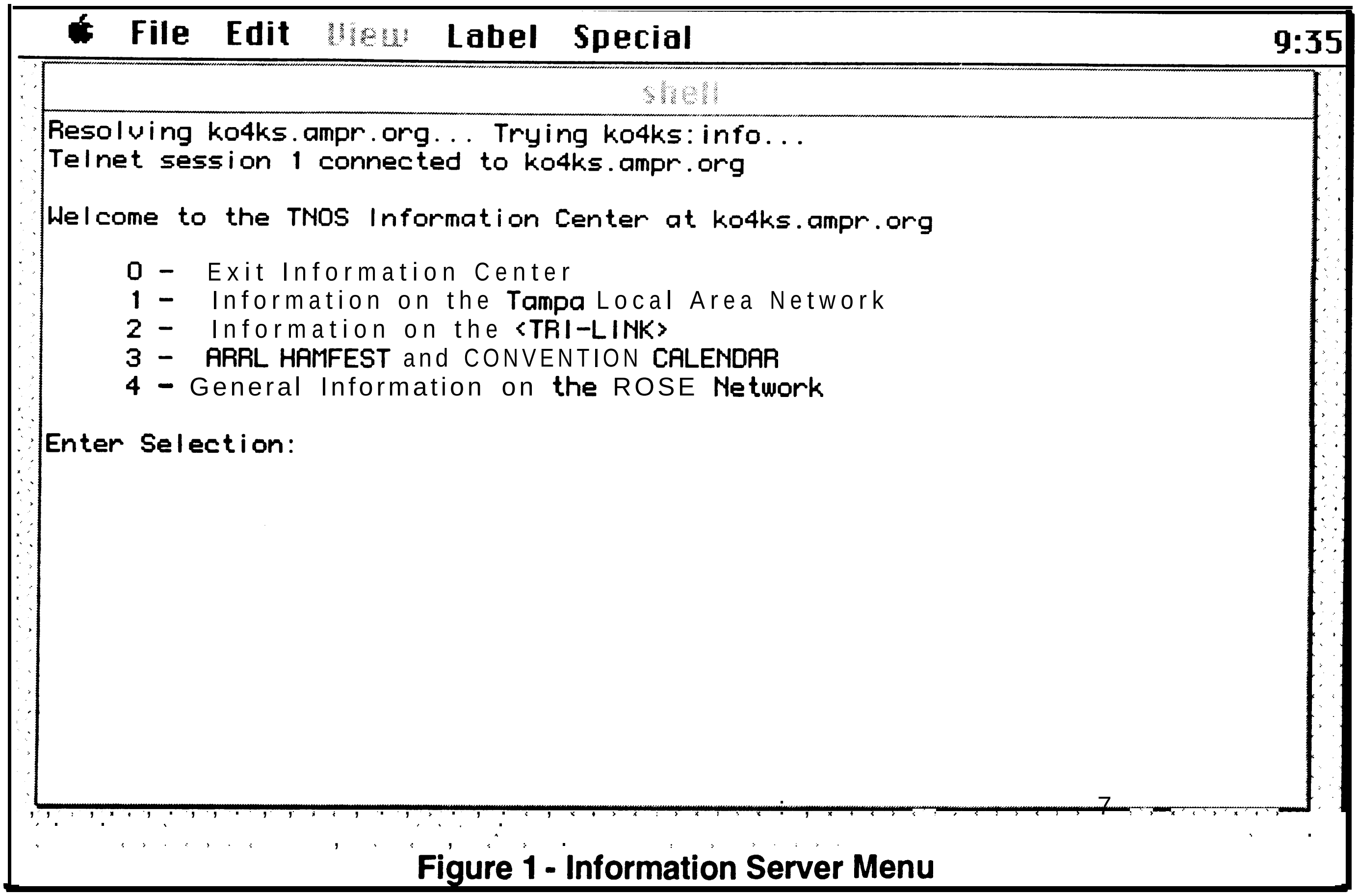
The 'subdir' is the name of a sub-directory within that directory. This allows you to define one-line files that make nested sub-menus.

Table 1 lists the many filenames in an Information Server directory and their title lines. Figure 1 shows how the user sees this information.



Filename	Title Line
tpalan.tut	~ tpalan Information on the Tampa Local Area Network
tri-link.tut	~ trilink information on the <TRI-LINK>
hamfests.tut	~ hamfests ARRL HAMFEST and CONVENTION CALENDAR
rose.tut	~ rose General Information on the ROSE Network

Table 1 - Filenames and their Title Lines



```
Resolving ko4ks.ampr.org... Trying ko4ks:info...
Telnet session 1 connected to ko4ks.ampr.org

Welcome to the TNOS Information Center at ko4ks.ampr.org

     0 -  Exit Information Center
     1 -  Information on the Tampa Local Area Network
     2 -  Information on the <TRI-LINK>
     3 -  ARRL HAMFEST and CONVENTION CALENDAR
     4 - General Information on the ROSE Network

Enter Selection:
```

Figure 1 - Information Server Menu

ANATOMY OF A SCRIPT FILE

The Tilde character ("~") is treated specially at all time. All lines that do NOT begin with the tilde "~" character are simply sent to the user after a few simple substitutions. All substitutions begin with the tilde character and serve as shortcuts for variable data fields. The special control sequences in text lines are:

~c	replaced by the user's callsign	~i0	replaced by value of index 0
~h	replaced by the name of the host computer	~i1	replaced by value of index 1
~d	replaced by the current date	~i2	replaced by value of index 2
~t	replaced by the current time	~i3	replaced by value of index 3
-0	**replaced by variable string 0**	~i4	replaced by value of index 4
~1	replaced by variable string 1	~i5	replaced by value of index 5
~2	replaced by variable string 2	~i6	replaced by value of index 6
-3	replaced by variable string 3	~i7	replaced by value of index 7
-4	replaced by variable string 4	~i8	replaced by value of index 8
~5	replaced by variable string 5	~i9	replaced by value of index 9
-6	replaced by variable string 6	~b	replaced with a bell character
-7	replaced by variable string 7	~~	replaced by a single '~' character
-8	replaced by variable string 8	~n	replaced with a newline character
-9	replaced by variable string 9	~u	un-terminate current line; remove <CR>

Table 2 - Text Control Sequences

The first four sequences allow you to place the caller's callsign, host computer name, or the current date or time into the text data sent to the user.

The next 10 sequences allow you to place a string variable into the text flow. There are 10 variable strings given for the script writer’s usage. Datafile text, user’s responses, and other data can be stored in variable strings and displayed with the "~0" through "~9" control sequences.

The next 10 sequences allow you to output a numeric variable. As with string variables, the script writer has 10 integer variables at his/her disposal, named "~i0" through "~i9".

The next two allow you to place a bell character or a tilde character into the stream. The last two sequences allow you to imbed a carriage return into a string, or mark a line to ignore its own carriage return at the end of the text line.

All lines that BEGIN with a '~' are treated as control lines by the Information Server. In all of the following tables, parameters marked with an astrick (*) can be either a literal number, an index counter (~i0 - ~i9), or an variable string (-0 - -9).

-<space>	defines a comment line; not printed or acted upon
~~ textline	a control line beginning with '~~' is treated as a textline with a '~~' control sequence at the beginning.
~b num*	output ‘num’ blank lines.
~m	prompt the user with “---MORE (*y/n)---“. The Server then waits for the user’s response. If it is anything other than ‘no’ (or 'n'), the Service File continues. If it is a 'n' response, the script ends. (see note in '~x')
~x	exit the Service File at this point. (Actually goes to label ‘exit’, if it exists).

Table 3 - Basic Control Lines

The “-<space>” control line allows you to imbed non-displayable, no operation lines that can be used to help document the script file. The "~b" control lines allow you to place a fixed number of blanks lines into the data flow.When the "~~" control line is used, the "~~" is simply treated as ordinary characters at the beginning of an ordinary text line.

The more control line ("~m") asks the user if he/she wants more, and if so continues. If a negative response is given, the script will do an implicit goto "~g" the label named “exit”. If “exit” doesn’t exist, the script is complete. You can also exit the script at any time with "~x".

You can easily take a regular text file, using these few control commands, and make a decent Service File. You should pace the display so that there is a user interaction or a “more” command for every screenful of data.

~l label	define a named label at this point in the Service file, named ‘label’.
~g label	goto the label line named ‘label’ and continue with the Server. The named label can be anywhere in the Service file.
~q prompt	query the user with the string ‘prompt’ and ‘(*y/n)‘. If the user responds with anything other than ‘no’ (or 'n'), the query status is set to ‘y’ (yes). This status is used by the "~y" and "~n" control lines.
~y label	if the query status is set to ‘yes’ by a "~q" or "~c" control line, then goto the label line named ‘label’ and continue with the Server. The named label can be anywhere in the Service file.
~n label	if the query status is set to ‘no’ by a "~q" or "~c" control line, then goto the label line named ‘label’ and continue with the Server. The named label can be anywhere in the Service file.
~v num prompt	send user ‘prompt’ string, get response from user, and assign the response to variable string ‘num’.

Table 4 - Query and Flow Control Lines

You can use labels very much like they are used in the C programming language and assembly language. A label is any text name that makes sense to you. Spaces are not allows in a label name. A label is defined with a “4” control line.

You can go to any label in a script file with the goto "~g" control line. The label can precede or follow the got0 line in the script file.

Simple quering can be done with the query "~q" control line. This command puts out a prompt string, asks for a yes or no reply, and sets the script’s response variable to either ‘y’ or ‘n’. The query variable is used for conditional gotos in the '~y' and "~n" control lines. These lines will go to the label given if the query variable matches (‘y’ for the "~y" command, 'n' for the "~n" command).

A different query variation exists with the "~v" (variable query) control lines. The prompt is sent to the user, and their response is placed in the string variable that matches "num".

-a num str	assign variable string ‘num’ with the string ‘str’. 'str' can also be any single special character sequence.
~p to fm s* l*	picks out a sub-string of a variable string ‘fm’ and places it in variable string to’. The sub-string starts at position ‘s (0 is the first position). The sub-string will be ‘l’ characters long, maximum.
~t num index	truncates (chops off) variable string number ‘num’ at position ‘index’ (0 is the first position).
~c nl n2 [lab]	compares two variable strings (n1 & n2). This sets query status to ‘y’ if equal, 'n' if not, for use with the "~y" and "~n" control lines. If the optional label ‘lab’ is given and the two strings are equal, then goto the label ‘lab’. The named label can be anywhere in the Service file.
~j nl n2 l* [b]	compares first 'l' characters of two variable strings (nl & n2). This sets query status to ‘y’ if equal, 'n' if not, for use with the "~y' and "~n" control lines. If the optional label ‘b’ is given and the two strings are equal, then goto the label ‘b’. The named label can be anywhere in the Service file.
~z var index	gets the length of variable string ‘var and places the length in index counter number ‘index’
~i#=[val*]	assign i# the value ‘val’ (or 1, if no ‘val’). The '#' is i0-i9. The ‘val’ is either a constant or a "~i#".
~i#+[val*]	add the value ‘val’ (or 1, if no ‘val’) to i#. The '#' is i0-i9. The 'val' is either a constant or a "~i#".
~i#-[val*]	subtract the value ‘val’ (or 1, if no ‘val’) from i#. The '#' is i0-i9. The ‘val’ is either a constant or a "~i#".
~i#?[val*][lab]	compare the value ‘val’ (or 1, if no ‘val’) to the value of i#. The '#' is i0-i9. This command sets the query status for use with the "~y" and "~n" control lines. If the optional label ‘lab’ is given and the query status is ‘y’, then goto the label ‘lab’. The named label can be anywhere in the Service file.

Table 5 - Assignment and Comparison Control Lines

You can assign an escape sequence or a literal string into a numbered string variable with the “-a” control lines. You can place a sub-string of one numbered string variable into a second numbered string variable with the "~p" control lines. The "~t" control lines allow you to truncate (clip off) a string variable at a certain point.

There are two control lines that can be used to compare string variables, "~c" and "~j". The first "~c" compares the entire string of each, while the "~j" command compares only the first part of the two strings. You can get the length of a string variable’s data with the "~z" control lines.

The "~i" control lines handle four different functions with integer variables. The character after the 'i' (0 through 9) indicates which numbered integer variable to use. The next character is the character that determines the function. The '=' assigns a value. The '+' adds to the current value of a variable. The '-'

subtracts from the current value. The '?' compares the variable to a set value and sets the query variable accordingly.

~f [filename] close any open I/O file and then create a new I/O file named ‘filename’. To close the current I/O file, give no ‘filename’. The query status is set to ‘y’ if the file open was successful.

-0 [filename] close any open I/O file and then open an old I/O file named ‘filename’. To close the current I/O file, give no ‘filename’. The query status is set to ‘y’ if the file open was successful.

~s seek to beginning of current I/O file.

-se seek to end of current I/O file.

~e [lab] test for an end-of-file condition on the current I/O file. This command sets the query status for use with the "~y" and "~n" control lines. If the optional label ‘lab’ is given and the I/O file is at the end-of-file, then goto the label ‘lab’. The named label can be anywhere in the Service file.

~w textline write the ‘textline’ to the current I/O file. The same special characters that apply to Text Lines apply to this ‘textline’, i.e. you can use the same control sequences.

~r num read the next line in the current I/O file into the variable string number ‘num’.

Table 6 - Data File Control Lines

An I/O datafile can be opened with either the "~f" or the "~o" control lines. Both commands close any open I/O file, and open the desired file. If no filename is given, these commands simply close any previously opened file. They differ in that the "~f" creates a NEW file and sets the query variable to 'n' if the file couldn’t be created or already existed. The "~o" command opens a new file and sets the query variable to ‘n’ if the file couldn’t be opened or didn’t already exist.

You can seek to the beginning ("~s") or end ("~se") of the data file. You can check for the end of file with the "~e" control lines. The query variable gets set to 'y' if at the end of the file. If a label is given on the control line and it IS the end of file, then a "goto" is executed, moving to the label line

You write to an open data file with the "~w" control lines. The rest of the line is treated the same as regular text lines, with imbedded control sequences allowed. You read from an open data file with the "~r" control lines. The next line in the data file is placed into the numbered variable string.

~u filename uploads (sends a text file) named ‘filename’ at this point in the script The file MUST be ONLY ASCII text. Any control lines in the upload file will be only displayed. The tutorial will continue, after sending the file, with the next line in theService file.

~k filename kills (deletes) a file.

~d user file delivers (mails) to ‘user’. The subject of the message will be the title line of the current Service File. The message is sent from the MAILER-DAEMON. The content of the mail message will be the contents of the ‘file’. The query status is set to ‘y’ if the message was successfully queued.

~? option a control line beginning with '~*' is treated as a request for information about ‘option’. The query status is set to ‘y’ (yes) or ‘n’ (no). This status is used by the "~y" and "~n" control lines.

Options available:

C	ANSI color graphics permitted
I	Connect was TCP/IP type

~! option a control line beginning with '~!' is treated as a request to change information about ‘option’. The status of the option is toggled on/off.

Options available:

C	ANSI color graphics permitted

Table 7 - File Utilities Control Lines

You can upload a separate text file with the "~u" control line. There is NO special processing of an uploaded file, so normal Information Server escape sequences are not examined, substituted, or acted upon.

You can use the “kill” command ("~k") to delete files on the host system’s disk. FOR THIS REASON, you should limit only those that you can TRUST to be permitted to add Information Server files on your system.

You can send the contents of a file in a mail message to a user with the "~d" control lines.

Information about a user can be examined and set using the "~?" and "~!" control lines. You can use these to check the user’s desire to use color. If the user came through the TNOS BBS and has ANSI color graphics mode on in the BBS, the color option will be on. Others will be initially set to off.

The username of the user is ONLY valid on starting a script if the connect was an AX25 connect, NOT a TCP/IP connect. You can find out if the user connected with IP using the "~? I” control line. If this is found to be an IP connect, you may wish to ask the user for a usemame and also ask them if they would like an ANSI display (if your script uses these features).

~* status	begins or ends a ANSI color block. The ‘status’ parameter is either ‘begin’ or ‘end’. ANSI sequences are “eaten” unless they are permitted for this user.
~% colorfile	displays a color file, if ANSI color graphics is permitted for this user. The query status is set to 'y' (yes) or 'n' (no) depending on whether or not the user permits ANSI. This status is used by the "~y" and "~n" control lines.
~ @ colorcode	changes current color set to “colorcode”, if ANSI color graphics is permitted for this user.

Table 8 - ANSI Color File Control Lines

The Information Server allows you to give users (that desire it) a more colorful display. You can change the text colorset, imbed ANSI control sequences in the text flow or add pre-saved ANSI color files at a particular spot in a Service File.

The "~*" control lines are used to mark the beginning and end of an imbedded ANSI sequence. The Information Server will remove KNOWN sequences from the data flow if the user has not selected ANSI graphics, but this is NOT without flaws. It is suggested that you use conditional looping based on the user’s color setting (using a "~? C" control line).

A pre-saved color file is inserted into the data stream with the "~%" control line, but ONLY is displayed if the user has color set to ON. Otherwise, the file will be skipped.

The current text colorset is changed with the "~@ " control lines. The “colorcode” is a two digit, hexadecimal value that determines the colorset desired. The first digit determines the background color and the blink attribute. The second digit determines the foreground color and the high intensity attribute. These color codes are listed in Table 9.

IMPLEMENTATION

While this is already implemented in TNOS, the same code could be easily integrated into other BBSs, other variants of NOS, or any other possible information provider.

There is a stand-alone application included in the TNOS distribution that allows ANY MS-DOS user to design/view Information Server files without needing to run TNOS.

1st digit	Blink	2nd Digit	Hi Intensitv	Color
0	Normal	0	Normal	Black
1	Normal	1	Normal	Blue
2	Normal	2	Normal	Green
3	Normal	3	Normal	Cyan
4	Normal	4	Normal	Red
5	Normal	5	Normal	Magenta
6	Normal	6	Normal	Brown
7	Normal	7	Normal	White
a	Blinks FG	8	Hi Intensity	Black
9	Blinks FG	9	Hi Intensity	Blue
A	Blinks FG	A	Hi Intensity	Green
B	Blinks FG	B	Hi Intensity	Cyan
C	Blinks FG	C	Hi Intensity	Red
D	Blinks FG	D	Hi Intensity	Magenta
E	Blinks FG	E	Hi Intensity	Brown
F	Blinks FG	F	Hi Intensity	White

Table 9 - Color Code Table

SAMPLE SCRIPT

What follows is a sample Information Server scriptthat serves no REAL useful purpose, but does test (and serve as a syntax example) most of the Information Server features.

```
                    First  Test  Tutorial

-1 loop
This is the first test at '~h'.

Now we will test the MORE function....
~b 3
~m
~b 3
It's time to test the QUERY function....
~b 2
~q Loop back
~y loop
~b 3
Tired of looping, huh!
One more test of the QUERY function....
~b 2
~q Loop back
~n noloop
~g loop
-1 noloop
~b 3
Really tired of looping, huh!

Time to test variables....
~b 3
~v 1 Enter variable #1 (at least 5 characters)

The variable entered was ~1!
~i9=4
~p 6 1 0 ~i9
The first four characters were '-6'. ~u
~z 1 4
```

```
The length of the string was ~i4.
~t 1 4
The truncated first four characters are '~1'.
Your call sign is ~c.
-a 4 This is variable #4
--Variable number 4 is '~4'.

~v 2 Enter your call sign
~v 3 Enter your call sign again
~b 2
~c 2 3 equal
They were NOT equal-b
~g next
-1 equal
They were equal and they were '~2'.~b
-1 next
~j 2 3 2 nowokay
Even the first two characters didn't match!
~g skipit
-1 nowokay
The first two characters matched!
-1 skipit
~m
~f /newtest
~w Call sign is '~c' - Variable 4 is '~4'.~n
Writing the following to disk:
'Call sign is '~c' - Variable 4 is '~4'.'~n
~f
-0 /newtest
~r 5
Variable 5 is '~5'.
Rewinding......
~s
~r 6
Variable 6 is '~6'.
~f
~k /newtest
~n xx1
Deleted temp file
~g xx2
~l xx1
Couldn't delete temp file-b
-1 xx2
~b2
The date is ~d.~nThe time is ~t.
~m
~ now for the index counters test
~b 2
Here is the equivalent code for "for (k=1; k < 10; k++)"
~i1=
-1 indexloop
The current value of index1 is -il.
~i1?10
~i1+
~n indexloop
~i2=~i1
The copied index from index1 to index2 is ~i2.
~m
~b2
And here is the equivalent code for "for (k=10; k; k--)"
```

```
~i1=10
-1  indexloop
The current value of index1 is -il.
~i1?1
~i1-
-n  indexloop
-1 lasttest
~b2
~v 9 Enter in a number from 1 to 4
It was ~u
~i9=~9
~i9?1 was1
~i9?2 was2
~i9?3 was3
~i9?4 was4
-1 invalid
not one of the numbers asked for! Your answer was-u
~i9?0
~n badone
 either zero-nor started with a non-digit! ~u
~g reloop
-1 badone
 ~i9! ~u
-1 reloop
Try again!-b
~g lasttest
~
-1 was1
One!
~g through
-1 was2
Two!
~g through
~l was3
Three!
~g through
-1 was4
Four!
-1 through
~m
Your current CONFIG.SYS file is....
~u /config.sys
~m
I'll now CONFIG.SYS to you in a mail message
~d ~c /config.sys
~n lastoops
Sent successfully!
~g lastout
-1 lastoops
Couldn't send it!
-1 lastout
~b 2
~q Ready to exit
-n loop
~b 3
-1 exit
Goodbye, come back and see us!-b
```

Listing 1 - A Sample, Illustration File

NETWORKING ANSI COLOR GRAPHICS

by Brian A. Lantz/KO4KS
TPALAN Tampa Local Area Network

BACKGROUND

A typical network supports two kinds of users; regular users and super-users. The regular users provide the NEED for the network. Without those who wish to USE the network, the network is useless! It doesn't matter if it is ROSE, NETROM, TCP/IP, or SplatRouter-2000; if there are no users, there is no need.

The second class of users, super-users, try to MEET the needs of the regular users. There are usually BBSs, but can also be information providers, Internet mail routers, non-radio wormholes to distant places, and any other resources that might be needed or desired.

While both classes of users should be given equal status pertaining to the use of network resources, the regular users, obviously, should be given special consideration. Their needs and desires should be constantly evaluated by the local administrators of the network, as well as by the writers of packet software.

The users in the early days of packet, the users had low expectations, since this was all a new experience. The users of 1993 have developed a desire to do more. They want new capabilities. They want to be able to do things more like landline networks. This brings us to ANSI color graphics.

This paper will briefly describe the needs, the shortcomings, the facts, and the solutions to making our networks support ANSI color graphics.

THE NEEDS

ANSI color graphics might seem to some to be only unnecessary fluff. In fact, they are correct! But so is almost all non-emergency, Amateur communications. In Amateur radio, perceived needs determine what is necessary.

The landline BBS world has for many years greeted its users with ANSI color text and graphics. While this doesn't change the content, it does make it ***seem*** friendlier and nicer. This environment has led to many public domain programs for creating ANSI color portraits, allowing the user to express his/her self with crude but expressive tools.

Once this budding artist has a creation that he has developed a deal of pride over, he now wants to SHARE the portrait with his peers. Acceptance in the ANSI art galleries is very important to those who have labored so long and hard.

So, the artist gleefully posts his creation in a message on the packet BBS system and sends messages to all his/her friends to tell them to check it out. What follows has broken lesser men, caused nations to crumble, and cancelled many good TV series.

The friends all send back messages saying that SOMETHING HAPPENED to the creation and it was not received as it must have been sent!

THE SHORTCOMINGS - BBS CHARACTER HANDLING

The current BBS software in use have totally different features and functionality, but they keep the message handling portions compatible. They might have OTHER WAYS to do these features in addition to the standard (i.e. SP, SB), but the interface for sending mail IS consistent.

Most of the BBSs have one or more control characters that can be used to abort the message, disconnect, etc. These character fall in the range of characters with values from 0 to 31. These characters (with the exception of the carriage return and line feed) are not usually used in ANSI color files. Because of this, it does not SEEM like there is a problem.

The problem starts, though, with ANSI graphic characters, which fall into the range of values 128 through 255 (characters with the most significant bit set). Some BBS software cannot handle characters in this range. They either bit-bucket these characters, choke on them, or strip off the eighth bit.

While most BBSs don't have a problem with MOST of these characters, a few have a problem with at least one of them, the character value 255 ($FF). The reason for this is-that character oriented libraries usually return a value of - 1 to indicate an error or the end of file. If you are: using, for example, the C programming language, and define the variable that you are filling with the next character to be a SIGNED CHARACTER (the default) rather than an UNSIGNED CHARACTER, that -1 looks the same as a 255 that has been sign-extended.

THE SHORTCOMINGS - TNC CHARACTER HANDLING

TNC firmware contributes to the problem, in most cases. Each manufacturer has commands that allow the 8th bit to be stripped from data coming in. These need to be properly set by each user.

In addition, most TNC firmware pre-processes the data that gets placed into its Mini-BBS. This pre-processing CAN corrupt the data.

THE FACTS - ABOUT ANSI CONTROL SEQUENCES

The biggest problems come from improper perception and inefficient packaging of the ANSI sequences within a packet.

While this is not the place for a complete dissertation of the details of how ANSI control sequences are handled, a little explanation is necessary for those who have not dealt with them before.

An ANSI control sequence starts with the ESCAPE character (27 decimal, $1B hexadecimal). This is followed by the left bracket ('[') character. All ANSI sequences begin with these two characters.

The length of an ANSI sequence is variable. After the beginning sequence, you may have zero or more numbers (in ASCII) separated by semi-colons (';'). Each number can be from 0 to 255. These numbers are the parameters for the ANSI command.

An ANSI sequence is concluded with a single alpha character (A-Z or a-z). This command code determines the function that this ANSI sequence is instructing to occur. A good source of reference for most of the normal ANSI sequences is the MS-DOS version *5.0 User's* Guide *and Reference,* pages 593-600 (ANSI.SYS).

```
$1B                    [    1          40    H
```

EXAMPLE 1 - Sample ANSI control sequence

In the example, the "ESC [" starts the sequence, it has two parameters (1 and 40), and the ANSI command code is "H". The "H" command sequence is a cursor positioning sequence. This particular example moves the cursor to line 1, column 40 of the display. Note the semi-colon between the two numbers.

Since all of these characters can be generated from a keyboard, the ANSI standard helps keep from generating FALSE sequences by placing an inter-character timer into the ANSI parser. Any pause between the beginning ESCAPE character and the terminating command code causes the parser to treat the interrupted sequence as an ordinary sequence of characters.

This gets complicated by packet radio, where the first portion of the sequence could be at the end of one packet, and the end of the sequence being found in the next packet. They could be separated by several seconds, far long enough for the ANSI parser to determine that it is not to be treated specially.

While there are some existing problems with the handling of 8-bit characters, most of the ANSI woes of the users are because of this broken sequencing.

THE SOLUTIONS

First, BBS and TNC authors, get into your code and find out what it is REALLY doing with characters that have the 8th bit set. Re-think your defaults in regards to these. The majority of the users are NOT using dumb terminals that need the 8th bit stripped. 8th bit transparency should be the default. Always treat your data buffers as UNSIGNED CHARACTERS. Try to make as few assumptions and limitations on the data as you can.

Second, TNC Mini-BBSs should store 8th bit characters correctly. By the time of this conference, PacComm should already be shipping release 3.2 of our Amateur firmware, which supports color file messages in the PMS.

Third, BBS authors should consider adding the few lines of code needed to properly packetize ANSI sequences. In TNOS all I had to do was check the queued data size when I came across an ANSI sequence. If the size was above 200 characters of data, I would flush the packet data BEFORE starting the ANSI sequence. The result is that the ANSI sequence begins a new packet and is never broken in the midst. That one simple change means the difference between a perfectly displayed creation, or a horrible fake.

CONCLUSION

While many of us are ready to start tackling digital voice across a network, some have much simpler needs. And while it may seem minor, this is the first step to such bigger hurdles. We can’t begin to pass other more complex data forms over the network, if we can’t work together on this one.

The Protocol Wars are (hopefully) over, and one things that I hope that we’ve all learned from them is that different people approach similar needs in different ways. The Vulcan edict IDIC stands for “Infinite Diversity in Infinite Combinations”. Our differences make us (collectively) better, when we can work together and learn from each other.

Brian A. Lantz
Manager of Software Development, PacComm Packet Radio Systems
President, Tampa Local Area Network
Packet: KO4KS@KO4KS.#TPA.FL.USA.NA
Internet: brianlantz@delphi.com

Interfacing between ROSE and TCP/IP

by
Thomas A. Moulton, W2VY
Radio Amateur Telecommunications Society

INTRODUCTION

The goal of the RATS Open System Environment (ROSE) X.25 Packet Switch from the beginning was to develop a radio network based upon standard protocols. It was also required that it provide reliable and completely **transparent** communications to all users of the AX.25 protocol.

For over a year now ROSE has properly supported passing AX.25 frames of any protocol type (PID). This was just the first step taken by ROSE in providing fully transparent services to **TCP/IP** users.

LEVEL OF SUPPORT

For our purposes we define support to mean that communications through each other is completely transparent. We should strive to provide services for the normal modes of communication. There also may be points where some additional information could be displayed to help give third parties insight into the type of connection and/or source and destination. It should be noted that features used to provide support, in many cases, are general features, not special features added to make a specific configuration work.

ROSE SUPPORT OF TCP/IP

In order for ROSE to support **TCP/IP** it must pass IP frames transparently. This can only be evaluated at the end points of a ROSE connection, as ROSE is only required to reassemble the original frame that was sent into the network.

VC Mode Connections

ROSE currently only supports IP connections in VC mode and provides two classes of service, unreliable and reliable mode. The only **difference** in these two modes is the action that is taken when data *might* have been lost. In reliable mode the connection is simply cleared. In unreliable mode all the data queued within the network is discarded and a ***** Call Reset ***** message is sent and then normal data transfer is resumed. The message is sent with PID=FO which would cause NOS to divert the connection to the BBS interface, ROSE will only sent this message if the normal data also has **PID=F0.**

When using unreliable mode the ROSE Switch **generates** two informational messages **Call being Setup** followed by a **Call Completed** or ***** Call Clearing ***** which are sent with **PID=F0**, this also causes the connection to be diverted to the BBS interface, even though: NOS is in the process of establishing this connection. These messages are sent before any user **data has** been sent, so the ROSE switch doesn't know what the PID of the data will be.

The net result is that NOS users are restricted to using ROSE reliable mode and NOS VC mode. One problem exists in this configuration, if the ROSE network detects a situation where data *might* have been lost it will tear down the connection, which will need to be restarted manually, since in NOS the VC mode handler does not attempt to reestablish a **VC** connection that fails.

Datagram Mode Connections

In the future ROSE will support **datagram** service, where **UI** frames will be forwarded through the network based upon the network address in the digipeater field. This will also allow for ARP broadcasts to specific ROSE user ports if the digipeater path can be changed on the fly. This is not a severe limitation, as there will be specific RF channels that have concentrations of **TCP/IP** users.

This will be a general feature independent of the actual **PID** in the UI **frame.** ROSE will manage an internal virtual circuit and maintain an idle timer that will keep a circuit open only while it is being used. This will reduce the number of X.25 Call establishments that will be required to pass the data.

TCP/IP SUPPORT OF ROSE

A ROSE switch must be able to establish an inter-switch AX.25 connection through a collection of NOS nodes. What is needed is an **AXIP** type of feature that is initiated by specifying a single digipeater field. Presently the **AXIP** feature is activated by digipeating through the NOS **callsign** followed **by** an alias that indicates the network address that the frames will be forwarded to. This is similar to the method ROSE uses to establish connections.

An additional extension to the AXIP feature would be to have the AX.25 Level 2 locally significant. This would not effect the actions of ROSE, but it would reduce the IP traffic, since the idle RR's would not be sent through the **IP** network. Then the only thing that would generate IP traffic would be an X.25 packet.

ADDITIONAL FEATURES

It would also be desirable for each environment to become aware of the existence of the other. The ROSE Switch USERS application will display the message **"TCP/IP** User" instead of **"AX25L2** User" for the entry and exit points of a connection through the network. It would also be useful if NOS would capture and display the ROSE network address for connections terminating at a NOS node. NOS should keep track of any of the PID=FO messages, which indicate Reset or Clearing cause code. These could be used to trouble shoot network problems.

TNOS - Tampa NOS

There is work being done by Brian Lantz, **KO4KS** on a more ROSE aware version of NOS. Tampa NOS (TNOS) eliminates most of the problems discussed. This version has changes in the following areas:

- Ignores PID=FO in VC mode **connects**
- Clean up for Dynamic Routing
- Specific IP Only **Callsign**
- Specific AX.25 Only **Callsign**
- Conference users list included each station's ROSE Address

With these changes the operation of **TCP/IP** is very predictable and will allow NOS users to use either reliable or unreliable mode ROSE connections.

CONCLUSION

If we all take a little time to make some simple changes in our software it can make the operation much simpler for all the users of our systems. It is not simply a matter of implementing the features needed to support every other system out there, but one of examining the general features needed to address the needs of many systems. I would like to thank Brian Lantz, **KO4KS** for his efforts in making **TNOS.** RATS will have the latest version available to those who want it.

Packet Radio in Disaster Situations

Robert Osband, N4SCY

PO Box 6841
Titusville FL, USA 32782
+1 407 383-1637 Voice
+1 407 267-4141 Fax
N4SCY @ N4ZIQ.FL.USA.NOAM Packet

Abstract

Packet Radio, a mode of data communications over amateur radio frequencies, was seldom used during communications operations following Hurricane Andrew. This paper will detail the situation as found by a user who went there to help, and proposed suggestions for future disaster recovery packet radio operations.

Introduction

I volunteered to work in an Oakland Red Cross Shelter for one 8- hour shift 5 days after the Earthquake hit in 1989. When Hurricane Andrew hit South Florida, I was living in Clearwater on Florida's West Coast. With 8 hours of "time in grade" (more actual disaster experience than any other amateur in the county), I was asked by the Pinellas County EC (Amateur Radio Emergency Coordinator) to lead a team of radio operators who responded to a call for operators sent by the ARRL Assistant Director for South Florida. The EC issued ARES (Amateur Radio Emergency Service) credentials to those who didn't yet have them, and off we went. ARES was founded by the ARRL (American Radio Relay League - the national organization of ham radio operators) which has a Memorandum of Understanding with the Federal Emergency Management Agency.

We were loaded with radios and equipment for VHF, UHF, and HF operation, including packet terminals, and computers for PBBS operation. When we arrived at the Dade County EOC (Emergency Operations Center) we were told that operations would primarily be 2 meter voice operation. A terminal and a TNC were set up in one corner of the EOC ham shack, but was not very active.

Our volunteers were deployed in two person squads to food diistribution centers, shelters, and a government headquarters building in Homestead. One evening, after my food distribution site had shut down for the day, I pulled out my packet gear, and set up for some "recreational computing." I consider packet to be my primary mode of operation, and it is the reason I got my ham license in the first place.

Operating packet in the affected area

I set up my Radio Shack Model-100 computer, PacComm MicroPower- TNC (Terminal Node Controller - the device that acts as a “radio modem”), and an Icom IC-2AT HT (Handy-Talky, or “hand held”) radio. As I connected the cable from the TNC to the radio, I did not push the plug all the way into the speaker jack, allowing a signal to go to the TNC, but not cutting out the speaker. This let me hear what (if anything) was on the air.

The first thing I tried to do was get on the frequency given for the Dade EOC, and connect to the packet station. No luck. I then tuned to 145.01 MHz in the middle of the major 2 meter “packet band” (144.91 to 145.09 MHz). I started reading a magazine, and every few pages, I would retune the radio. At 145.09 I heard the familiar "Brraaaaapp" of a packet transmission. I looked at the computer screen, and saw a beacon from the MIA7 NetROM type node from (I assumed) Miami. Ordinarily, the "7" would mean that the node was usually found on .07, but I figured that it was probably put on .09 for some reason due to the emergency. I tried to connect to it.

While timing out, I heard another "Brraaaapp" go by. After my Connect Request to MIA7 timed out, I typed "MHEARD" to see what callsign had gone by. I tried to connect to MIA7 via this new station, and Bingo! "*** Connected” appeared on my screen. I quickly typed "C HWD" to instruct the node to Connect me to the Hollywood FL Node by whatever convoluted route of IAN (Local Area Network) and Backbone circuits the Node decided to use. "### CONNECTED” came back to my screen. Now I knew where to go, since I’d been briefed by our PBBS (Packet Bulletin Board System) Guru back in Clearwater.

I then typed "C W7LUS" to Connect to the W7LUS PBBS. This board had a direct HF link back to the W4DPH PBBS in Clearwater. I sent two pieces of message traffic that night. One was a report of our deployment to our EC. The second was the only piece of NTS traffic I sent the entire week we were in the Homestead area. It said “Hi Mom, I’m OK”. While my own family was safe, they knew I was coming into the affected area, and I didn’t want them to worry.

Observations

It was amazing that for a communications based hobby, there was such a lack of communications. (My observation of Phone Phreaks of my acquaintance is that they are people who do not know how to communicate well with people, so they get into the technology of telecommunications to compensate.) I didn’t learn until after ! returned home that packet messages could have been left on the W7LUS PBBS, and they would have automatically been routed to the Mailbox in the TNC at the EOC. Lists of required items could have been dispatched by computer, and printed out at the EOC, or forwarded to the food distribution centers.

Our messages were sent by voice, and hand written onto pads, and dispatched into "The Pit”, or Operations Center of the EOC. The agency the message was addressed to would receive the message, and that agency would handle the contents of the message from there. One major drawback seemed to be that messages coming from the ham shack were not handled with dispatch, until things seemed to get critical (either through shortages, or lack of support personnel). My opinion is that people and agencies are not used to message based communications systems any more.

Our message handling skills are based on the telegraphic message systems which had their heyday in the 40's and 50's, and declined with the affordability of the telephone into every home and office, to the point where the Western Union Telegraph Company seems to only have the "telegraphic money order" business left. People no longer expect that a message is "authentic" unless it came up through a "proper chain of command". Ham radio did not appear to have that "level of authenticity" to our user community. Our users did not understand that any message sent by an operator had (or should have had) an "original" of any message written down in a log, with the signature of the originator on file should followup after the event be required.

Had packet radio been in active use, there would have been that one magic element added to the message. It would have been a printout, and it is a modern myth that anything printed out by a computer is Gospel. A printout would have that certain "cachet" that might have gotten a message through. Particularly a list of needed supplies in a computer format.

Recommendations

With laptop computers becoming more common at reasonable prices, teams planning to provide packet radio capability for their Quick Response Volunteers (The acronym turns me on, while the acronym for Quick Response Team turns me off) should have with them TNC's with EPROMS for NetROM type operation. While I personally prefer ROSE switch type operations, configuring a ROSE switch in a disaster area seems impractical. NetROM type nodes are self- identifying, and configure themselves into a network. The thing is that if a QRV team arrives with the intention of providing networking capability, they should also have UHF equipment with them so that their VHF Node can operate their own Local Area Network in the affected area, and be linked back into a UHF backbone to link LANs in the area.

PBBS operators who are operating in or near the affected area should have log-in bulletins pointing to message numbers and text files for users to read, such as:

"Hello N4SCY, Welcome to the W4XYZ PBBS. For instructions on sending messages to the EOC, read message 12345. For a map of the LAN in the emergency area, type 'D EMERG.MAP'. To volunteer to help, call Net Control on the 146.64 repeater."

In the Hurricane Andrew operation, we operated all day, and our sites shut down in the evenings. Our teams usually slept on-site, and near their equipment. This meant that in the evenings, a voice net could have been called for operators to swap information, allowing our people to have insight as to the situation at other locations. The Pinellas County squads did this informally among ourselves, and off the regular net frequencies.

Besides keeping our team in touch with each other, this information on the status of other sites was usually invaluable to our site managers during discussions the next day.

Another thing that would have been very helpful would have been another command in the TNC. Besides the MHEARD list, a DHEARD (Digipeaters HEARD) would have been invaluable. The J L command (Just heard, Long) command in KaNodes is similar to what I'm looking for here. When not connected to a station and monitoring a frequency, an Asterisk shows up next to the call sign of the station you actually heard, as in:

N4SCY > W4XYZ > K4ABC* > ORL7: < UI > This is Ouie in Titusville.

In the example, the packet sent by N4SCY was heard when it was retransmitted by K4ABC. Since K4ABC was the Digipeater heard by my station, it would then appear in the proposed DHEARD list. This would allow a user in a disaster area to quickly find out what digipeaters are active, even when he's been operating, and too busy to monitor activity to find the digi's himself. They say, "If you can't hear 'em, you can't work 'em". Well, if you don't know you've heard them, you can't work through them.

Conclusion

There are many things that the packet community can do to help provide communications for our communications clients during a disaster. The users need to be made aware of the capabilities, and demonstrations need to be made to Emergency Management officials so they know what those capabilities are, and how useful they can be to disaster relief operations. Before all this can happen, the operator community needs to be up speed on what to bring to a disaster, and how to use what the operator brings.

AR

de Ozzie N4SCY @ N4ZIQ.FL.USA.NOAM

NNNN

THE *HUB 5/29* IP ROUTING EXPERIMENT

by Paul Overton, GOMHD @ GB7MHD (paul @ g0mhd.demon.co.uk)
and Ian Wade, G3NRW @ GB7BIL (g3nrw @ dircon.co.uk)

This paper summarises the disadvantages of default IP routing, which often leads to total traffic loss when attempting to forward over long distances. This is followed by a description of how to set up a hub routing scheme that overcomes these problems. An experimental scheme along these lines has been successfully implemented in Regions 5 and 29 in the UK (hence the Hub *5/29* in the title of this paper).

Introduction

For the purposes of allocating IP addresses, the UK is split into geographical regions based on county boundaries. IP addresses are of the form 44.13 1 *.RR.SSS*, where RR is the region number and *SSS* is a station address within the region. Regional coordinators typically issue addresses in ascending numerical order on a first-come first-served basis, with little consideration for network planning. It doesn't matter where in the region a station is located, or who its neighbours are, or how to forward traffic within the region. Each new station gets the next available address, and that's that.

With the rapid increase in TCP/IP activity over the last year or so, maintenance of IP routing tables has become a real problem. Unless users regularly update their hosts files, and keep an eye on how the network is changing around them, it soon becomes unrealistic to maintain suitable IP routing tables. As a result, reliable message forwarding and file transfer over more than two or three hops have now become virtually impossible for most people.

It thus became evident that two things were urgently needed to overcome these difficulties:

1. a means of keeping users up-to-date with network host addresses.
2. a simple algorithm for setting up the IP routing tables.

The first of these requirements has been met with the introduction of Domain Name System (DNS) servers. DNS servers hold a more-or-less complete list of AMPRnet host names and IP addresses, so that ordinary end users don't need to maintain their own *domain.txt* file. Whenever a user says **telnet zzz** or **ftp** zzz, their system interrogates the local DNS server for the IP address of station **zzz**, and thereafter uses the address provided by the server to make contact with zzz. The big advantage of this scenario is that ordinary users only need a *very* short *domain.txt* **file,** and they automatically track any changes to **zzz**'s IP address.

The second requirement, to provide a simple algorithm for setting up the IP routing table, is the main subject of this paper.

The problem

In principle, IP routing is very simple. All you need to do is set up the routing table to send traffic in roughly the right direction, hoping that the next station down the line will do the same. And the next station does the same. And the station after that does the same....

Trouble is, in addition to specific routing table entries for known destinations, most people have a default catch-all routing table entry (set up by the command **route add default tnc0**, for example) to forward any traffic which doesn't match the specific routes. Because the default route doesn't include a gateway, traffic for unknown destinations will simply be forwarded to any station which happens to be in immediate radio range, and may therefore go off in completely the wrong direction. Eventually, unless you're lucky, it's probable that the traffic has nowhere else to go, and will finally evaporate in that big bit bucket in the sky!

To prevent this happening, what you need to do is route any traffic addressed to unknown destinations through specific gateways. The trick is to define an ideally small set of **route add** commands which will handle this scenario.

This is where the new hub organisation comes in. With hubs, we have a structured, well-defined network which is easy to understand, and where everyone knows exactly how to forward traffic on to the next station in the network. Routing tables are very short and simple, and assuming that everyone follows the rules it's possible to forward traffic over large distances with very little effort.

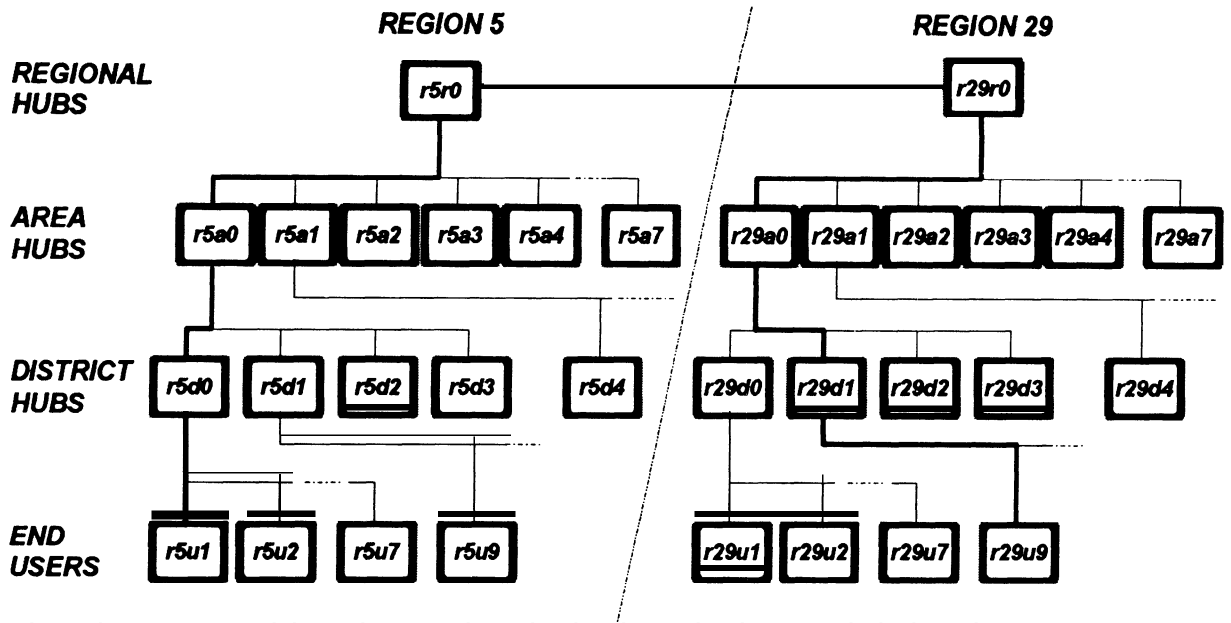


Fig 1: The UK IP network in Regions 5 and 29, showing the Regional, Area and District Hubs. The thick black line shows the route between End User stations *r5u1* and *r29u9.*

How do hubs work?

In the United Kingdom, Region 5 covers the counties of Bedfordshire and Northants, and Region 29 covers Cambridgeshire (see Fig 1 above). In general terms, the Region **5/29** network looks like the following:

Starting at the top of Region 5, the Regional Hub station is called **r5r0** (i.e. region **5** regional hub 0).

Underneath the Regional Hub there are 8 Area Hubs: **r5a0** to **r5a7** (region **5** area hub 0 to region 5 area hub 7).

Underneath each Area Hub there are 4 District Hubs; for example, District Hubs **r5d0** to **r5d3** sit under Area Hub **r5a0,** District Hubs **r5d4** to **r5d7** sit under **r5a1,** and so on.

Finally, there are 7 end users per District. That is, users **r5u1** to **r5u7** are in District **r5d0,** users **r5u9** to **r5u15 are** in District **r5d1,** and so on.

A similar arrangement exists for Region 29.

Sending packets from one region to the other

To see how the hubs work, let's follow the route that traffic takes from user **r5u1** in Region 5 to user **r29u9** in Region 29.

Starting at **r5u1,** the **traffic** passes first to its District Hub **r5d0,** then upwards to Area Hub **r5a0,** then to the Regional Hub **r5r0.**

Hub **r5r0** then forwards the **traffic** across to Region 29's Regional Hub **r29r0,** and from there it trickles down through Area Hub **r29a0** and District Hub **r29d1,** which forwards it to its final destination **r29u9.**

The Routing Tables

Nothing remarkable so far. Now let's look at what is needed in the routing tables of all the stations traversed between **r5u1** and **r29u9.**

At User r5u1: An easy one here. In this model, there is only one way for *all* **traffic** to go — upwards to district hub **r5d0.** So to set up the routing table at **r5u1,** all we need is:

```
route add default tnc0 r5d0
```

(assuming the interface name is `tnc0`)

That's the complete routing table.

In other words, the routing table for every end user consists of just one entry; **a** default entry which routes all **traffic** to the end user's District Hub.

At District Hub r5d0: At the District Hub we have to cater for two alternatives:

1. We forward **traffic** back downwards to another end user in the same district, or
2. We forward traffic upwards to the Area Hub.

To handle the first alternative, we could have 7 separate **route add** statements, one for each end user in the district. However, by choosing IP addresses carefully, we can reduce this to just one **route add.**

For example, we can allocate the users in this district a block of addresses in the range 44.13 1 .5. 1 to 44.13 1.5.7. Expressing these addresses in binary:

```
         44   .   131   .   5   .   x

r5u1: 00101100.10000011.00000101.00000001
r5u2: 00101100.10000011.00000101.00000010
r5u3: 00101100.10000011.00000101.00000011
r5u4: 00101100.10000011.00000101.00000100
r5u5: 00101100.10000011.00000101.00000101
r5u6: 00101100.10000011.00000101.00000110
r5u7: 00101100.10000011.00000101.00000111
      <---------- 29 bits -------->
```

Looking carefully at these addresses, we see that the first 29 bits are identical. It's only the last 3 bits which differ.

So to set up a routing table entry to forward downwards to these 7 users, all we need to do is set up a **29-bit** routing mask:

```
route add 44.131.5.0/29 tnc0
```

This means that all traffic addressed to any station whose address matches the first 29 bits of **44.131.5.0** is forwarded through interface **tnc0**. Because there is no IP gateway included in **the route add** command, the end users have to be in direct radio range of the hub.

And for all other traffic? All we need is a second **route add** command to forward it upwards to the Area Hub **r5a0**:

```
route add default tnc0 r5a0
```

Thus each District Hub needs just **two route add** commands: one command containing a **29-bit** mask to forward traffic down to other users in the same district, and a default command to send the remaining traffic up to the Area Hub.

At Area Hub r5a0: The story continues along the same vein. At the Area Hub, we must cater for two alternatives:

1. We forward traffic back down to another District Hub in the same area, or
2. We forward traffic upwards to the Regional Hub.

To handle forwarding back down to another District Hub, we again choose our IP addresses carefully. We've already seen that District Hub 0 (**r5d0**) handles the following end user addresses:

```
r5u1: 00101100.10000011.00000101.00000001
                       to
r5u7: 00101100.10000011.00000101.00000111
```

If we now give the District Hub **r5d0 the** address 44.131.5.0, its binary equivalent is:

```
r5d0: 00101100.10000011.00000101.00000000
      <---------- 29 bits -------->
```

That is, the first 29 bits of the District Hub address are the same as the first 29 bits of all addresses below the hub.

So, to forward from the Area Hub down to District Hub 0 and any stations 'below it, all we need is the command:

```
route add 44.131.5.0/29 tnc0 r5d0
```

Similarly, if we choose the following addresses for the remaining 3 District Hubs:

```
r5d1: 00101100.10000011.00000101.00001000
r5d2: 00101100.10000011.00000101.00010000
r5d3: 00101100.10000011.00000101.00011000
```

then all we need in the routing table for these District Hubs is:

```
route add 44.131.5.8/29 tnc0 r5d1
route add 44.131.5.16/29 tnc0 r5d2
route add 44.131.5.24/29 tnc0 r5d3
```

Once more we need a default route upwards to the Regional Hub for all other traffic:

```
route add default tnc0 r5r0
```

Thus each Area Hub needs just four **route add** commands for the District Hubs below it, and one default command to route remaining traffic up to the Regional Hub.

At the Regional Hub r5r0: We've reached the top of Region 5. Once again, we have to consider two alternatives:

1. We forward downwards to one of the 8 Area Hubs in the same Region, or
2. We forward across to the Regional Hub of Region **29.**

By now, the pattern is clear. To forward downwards, we need 8 **route add** commands, one for each of the Area Hubs. The Area Hubs cover the following end user addresses:

```
r5a0:   00101100.10000011.00000101.00000000   (44.131.5.0) to
        00101100.10000011.00000101.00011111   (44.131.5.31)

r5a1:   00101100.10000011.00000101.00100000   (44.131.5.32) to
        00101100.10000011.00000101.00111111   (44.131.5.63)

r5a2:   00101100.10000011.00000101.01000000   (44.131.5.64) to
        00101100.10000011.00000101.01011111   (44.131.5.95)

r5a3:   00101100.10000011.00000101.01100000   (44.131.5.96) to
        00101100.10000011.00000101.01111111   (44.131.5.127)

r5a4:   00101100.10000011.00000101.10000000   (44.131.5.128) to
        00101100.10000011.00000101.10011111   (44.131.5.159)

r5a5:   00101100.10000011.00000101.10100000   (44.131.5.160) to
        00101100.10000011.00000101.10111111   (44.131.5.191)

r5a6:   00101100.10000011.00000101.11000000   (44.131.5.192) to
        00101100.10000011.00000101.11011111   (44.131.5.223)

r5a7:   00101100.10000011.00000101.11100000   (44.131.5.224) to
        00101100.10000011.00000101.11111111   (44.131.5.255)

        <--------- 27 bits -------->
```

This time we're interested in the first 27 bits of the address. They specify the Area Hub number, **so** we need the following **route add** commands at the Regional Hub:

```
route add 44.131.5.0/27   tnc0 r5a0
route add 44.131.5.32/27 tnc0 r5a1
route add 44.131.5.64/27 tnc0 r5a2
route add 44.131.5.96/27 tnc0 r5a3
route add 44.131.5.128/27 tnc0 r5a4
route add 44.131.5.160/27 tnc0 r5a5
route add 44.131.5.192/27 tnc0 r5a6
route add 44.131.5.224/27 tnc0 r5a7
```

Finally, we need a **route add** command for **traffic** going across to Region 29:

```
route add 44.131.29.0/24 tnc0 r29r0
```

(plus, of **course,** similar **route add commands** for traffic going to other regions).

The big attraction of hubs

The **route add** commands listed above are the only such commands which are necessary to forward any and all traffic within, into and out of a region. This is what makes the hub scheme so attractive.

With the hub scheme, all you need to know is how to forward to your **next-door** neighbour. In turn, your **next-door** neighbour knows how to forward to his/her next-door neighbour, and so on.

Also, because the routing tables at the hubs are very small and will **only** require infrequent changes, version Xl-G (or later) of **TheNet** is an ideal candidate for building the hubs — with NET/ROM forwarding disabled, of course! You don't need a computer for such a hub, just a **TNC** with an Xl-G EPROM and a radio.

Another big advantage of hubs is that because the network is consistently structured and all routes are known by everyone, there is no need for routing update broadcasts. This means that we no longer have RIP or RSPF **traffic** clogging up the network; they just aren't necessary.

so, to **summarise**:

- End users need *one and only one* **route add** command.
- District Hubs **need** *just two* **route add commands**.
- Area Hubs **need** *just five* **route add commands**.
- Regional Hubs **need** *just eight* **route add** commands for the areas below it, plus one command for each other region it forwards to.

Don't be silly!

Question 1: **If** I can reach a station in my own district direct, is it really necessary to forward up to the District Hub and down again to my neighbour?

Answer 1: Of course not. There's nothing to stop you including more **route add** commands to handle these special situations. So your complete set of **route add** commands may look something like this:

```
route add r5u2    tnc0
route add r5u7    tnc0
route add default tnc0 r5d0
```

In other words, you can add special routes for all stations which you can reach directly. The only proviso is that it's entirely your own responsibility to maintain these routes, and that the rest of the network may be unaware of them.

Most important of all, you must *always* have a default route *which points to another station.* In other words, this is all right:

```
route add default tnc0 r5d0     ☺
```

but this won *'t* do:

```
route add default tnc0       ☹ <<< WRONG
                               (ngatewayspecified)
```

Question 2: What do I do if I live in Region 5 and want to talk to someone in Region 29 who happens to be just a couple of miles away across the county boundary. Surely I don't have to climb all the way up to the top of the Region 5 tree and then descend all the way down the Region 29 tree to the station I want to talk to?

Answer 2: Same as answer 1. So you may have an additional **route add** command like this:

```
route add r29u7 tnc0
```

Question 3: What do I do if1 live just outside Region **29** in Region 19, but can reach a Region 29 station direct? Do I have to forward my **traffic** towards the Region 29 Regional Hub?

Answer 3: Same as answer 2. Alternatively, it may make more sense for you to trade in your Region 19 address for a Region 29 address to improve connectivity, even though you don't actually live in Region 29. The regions are defined for administrative convenience only — radio waves don't respect county boundaries, and we're trying to build a workable network here, not to satisfy the bureaucrats or the pedants!

What you need to do

To participate in the hub scheme as an end user, this is what you need to do:

1. Get yourself a new IP address.
2. Set up your *domain. txt* file.
3. Set up your DNS client (optional).
4. Set up the **IP** routing table.
5. Disable routing broadcasts"
6. Set up the SMTP gateway for outgoing mail.
7. Set up the POP client to collect incoming mail.

The following sections describe these in detail. Several examples of commands are included — these examples apply specifically to PAOGRI versions of NOS, but they are almost certainly the same (or very similar) for other well-known versions as well. These commands will go in the NOS startup file (called *autoexec.nos* or similar).

Most of the examples contain the parameter `<District Hub>`. 'Where you see this, substitute the **IP** hostname-of your own District Hub. So, for example, where you see the command syntax **smtp gateway** `<Di strict_Hub>` and your District Hub is, say, **r5a0**, you **will** include the command **smtp gateway r5a0** in *autoexec.nos.*

1. Getting a new IP Address

To take advantage of the the new hub routing scheme, you will need a new IP **address.** Contact your local IP address coordinator for this.

2. Setting up *domain.txt*

Your address coordinator should be able to provide *you with* a suitable *domain.txt* file. However, you will only need this **file** if you can't access a DNS server.

3. Setting up the DNS Client

If there is a DNS **server** within two or three hops range, you should set up your station as a DNS client. This means that *you only* need a minimal *domain.txt* file in your own system.

For example, your *domain.txt* could be literally as short as this:

```
$origin ampr.org
r5u1      IN A 44.131.5.1 # my own call
r5d0      IN A 44.131.5.0 # my District Hub

          IN NS r5dns
r5dns     IN A 44.131.5.95 # Region 5 DNS server

loopback IN A 127.0.0.1
```

N.B. The name server can be anywhere — it doesn't have to be in your own region.

Then, to make your DNS client interrogate the DNS server, you'll need this command in *autoexec.nos*:

```
domain addserver r5dns
```

4. Setting up the IP Routing Table

To set up the IP routing table, all you need is a single **route add** command to forward all default traffic to your District Hub, plus possibly a handful of other **route add** commands for your immediate neighbours within direct radio range.

Thus, in your *autoexec.nos*:

```
route add default tnc0 <District_Hub>
route add r5u9 tnc0
route add r29u23 tnc0
```

where **r5u9** and **r29u23** are special entries for immediate neighbours.

Note that if you include such special routing entries, your neighbours must similarly set up special entries in their routing tables to point back to you. These special entries are not part of the hub forwarding scheme, so it is up to you and your neighbours to maintain them.

REMINDER: Most important of all, for the hub scheme to work, **the route add default** command *must* include a gateway.

5. Disabling Routing Broadcasts

Because the hub routing scheme has a welldefined and consistent addressing structure, it isn't necessary to broadcast IP routing table updates. Thus you should turn RIP and RSPF off, by commenting out **any rip** and **rspf commands** which you may have in *autoexec.nos*; i.e. precede these commands with a # character.

6. Setting up the SMTP Gateway

You will probably use the Area Hubs as SMTP gateways to forward your outgoing mail. Thus you should include **an smtp gateway** command in *autoexec.nos*, defining your mail gateway:

```
smtp gateway <Area_Hub>
```

7. Setting up the POP Client

If you don't want to leave your radio on all the time, you can ask your Area Hub sysop to hold the mail for you until you are ready for it.

To collect the mail, you'll need to set up your POP client, specifying the **popmail** server, the name of your mailbox on the server, and the POP account name/password. Thus in *autoexec. nos*:

```
pop mailhost  <popmail_server>
pop mailbox   r5u1
pop userdata r5u1 mypassword
pop timer     3600
```

The name and password which you give in the **pop userdata** command must match the entries in the POP accounts file that the Area Hub sysop sets up in his system.

As another example, if you and the Area Hub support POP3, you can include this command in *autoexec.nos* (all on one line):

```
popmail addserver <popmail_server> pop3 r5u1 r5u1
                                           mypassword
```

The first **r5u1** is the name of your mailbox at the server. The second **r5u1** and **mypassword are the** POP account name/password pair.

Once this is all in place, all you have to do to collect your mail is to give a **pop kick** command from the keyboard:

```
net> pop kick <popmail_server>
```

Contacting Region 5 and Region 29 stations from the outside

If you live outside the Regions 5 and 29, and want to communicate with somebody inside, there are three options:

1. If you are within direct radio range of the wanted station, set up your routing table for a direct connection. There's :no point in going up and down the hub hierarchy if you don't need to. As explained above, the wanted station should set up a similar entry to point back to you.
2. If you are reasonably close to an Area Hub (say, within two or three hops), forward all traffic for Region 5 and Region 29 to that hub.
3. If you live a long way from Region 5 or 29, forward your traffic towards the following hubs:

dunbbs [44.131.5.120] for Region 5 destinations

mhdbbs [44.13 1.29. **1]** for Region 29 destinations

For options 2 and 3, you'll need a couple of **route add** commands in *autoexec.nos*:

```
route add 44.131.5.0/24 tnc0 <gateway>
route add 44.131.29.0/24 tnc0 <gateway>
```

(The <gateway> is a next-door neighbour which can forward traffic towards the chosen hub).

How it has worked in practice

Summarising our experiences in Region 29 (the story for Region 5 being broadly similar), the situation at the inauguration of the new hub routing system was that three of the four proposed Area Hubs were in place and operating. The fourth Area Hub was to have many problems with software etc, but these were fixed after some effort.

Issuing New Addresses

The first major problem to tackle was that of issuing the new addresses and getting the routing fixed so that we could make use of the new structure. Region 29 started with about 40 listed users, all whom changed their addresses within three days of the hub system starting. The issuing of addresses was based upon the rules as laid out above, but also adding a sub-rule for other radio frequencies as well — this meant that addresses were issued on the basis of location and radio frequency.

Most users were surprised by the simplicity of the changeover. With a few exceptions, the hub system became operational within just two days of the starting date. Some of the more adventurous immediately tried to use the newly available routes, both across the region and some out of the region. Most routes worked well but some were a little slow.

The inter-region link between Regions 5 and 29 worked very well from day one.

Starting small

The hub system started with only two layers: Area Hubs and end users. This was mainly because there weren't enough stations around or there wasn't enough equipment available to set up District and Regional Hubs. Routing for the users was exactly as set out above. However, because there was no Regional Hub for either region, the routing between the Area Hubs in Regions 5 and 29 (and to other regions as well) was a little more complex. A consequence of this is that there can be no default route for any of the Area Hubs because there are routes in all directions linking with other regions.

Since the hub system started there has been a steady growth in the number of applications for addresses, It was not long before it became necessary to start the first District Hub. In order to prevent the District Hub from overlapping its radio coverage with that of the local Area Hub, it was (decided that the main user frequency of the District Hub would be different from that of the Area Hub. The inter-hub link is on 4m and the user frequency is on 2m, 50 kHz above that of the Area Hub. This works. well, but because of the increasing level of user traffic a higher speed radio link is now being planned. There are also plans for more District Hubs.

Below is an extract from the routing table at Area Hub **mhdbbs.ampr.org**:

```
#Region 29 routes from mhdbbs.ampr.org.

 route add 44.131.29.16/29 ax0

 #Area Hub 1
 route add 44.131.29.32/27 netrom 44.131.29.3 1
 route add 44.131.29.64/28 ax0   #VHF user block

 #District Hub 1
 route add 44.131.29.80/29 ax4 44.131.29.80 1
 #Additional 4 addresses for District Hub.
 route add 44.131.29.88/30 ax4 44.131.29.80 1

 route add 44.131.29.95 node     #route to DNS

 #User Subnets
 route add 44.131.29.96/30 ax5
 route add 44.131.29.100/30 ax3 44.131.29.100 1
 route add 44.131.29.104/30 ax3
 route add 44.131.29.108/30 ax5 44.131.29.108 1

 route add 44.131.29.112/29 ax1  #UHF user block
 route add 44.131.29.120/29 ax0  #2nd VHF user block

 #Area Hub 4
 route add 44.131.29.128/27 netrom 44.131.29.2 1

 #Area Hub 5
 route add 44.131.29.192/27 netrom 44.131.29.4 1
 #Route to Region 5
 route add 44.131.5.0/24 ax2 44.131.5.120
```

Some of the interlink routes use NET/ROM as the carrier. This has been done because these routes use the existing network. The use of NET/ROM ensures that the IP network still functions, even if only slowly, when the existing network is busy. Also, most of the NET/ROM routes use radio frequencies which are not generally used for user access.

Introducing the Mail Hopper

With greater distances now attainable, a new problem surfaced: how to move mail efficiently across the network. It was found that the normal system of point-to-point SMTP sessions was very slow owing to the number of hops that frames were having to go through. MX records were tried, but the disadvantage was that an MX record was required for each user at every point throughout the network.

So a new mail forwarding technique was adopted, called the ***mail hopper***. The hopper applies a route-related approach to mail forwarding, using IP addresses instead of hostnames. This required small modifications to the SMTP client code at each hub: after resolving the destination's IP address, the client now consults the routing table to discover the IP gateway through which the mail is to be forwarded. This simple change allows the mail to be forwarded entirely at the IP level to its destination, thus removing the overhead and maintenance problems of intermediate SMTP gateways. The hopper works very well in practice, and has significantly improved mail transfer times through the network.

POP mail

Because most users don't run 24-hour hosts, it was decided to implement the POP mail retrieval system throughout Regions 5 and 29 — it had been in use for some time already by a few users, but with the introduction of the hub system it has now become the normal method of retrieving private mail for almost everyone.

NNTP

The hub system has also presented us with an ideal mechanism for communicating with all users at all locations throughout the network, using NNTP. To do this, all the Area Hubs have now been set up as NNTP sewers and poll each other every hour. Initially the quality of the user software was not good for this function, but the situation has now been improved by the addition of third-party news readers like SNEWS and CPPNEWS. Again, NOS had to be altered to cope with this.

The NNTP service has significantly improved the quality of user support, as messages for help can now be seen and answered by all. This has turned out to be an unexpected benefit — before the introduction of easy-to-use NNTP services, the support burden for the network was shouldered by just two or three people, whereas now the level of expertise is spreading rapidly over the whole of Regions 5 and 29, and beyond.

The Future

There are now a number of other regions successfully using the hub system in the United Kingdom and slowly we are achieving real network connectivity. We hope to continue this process so that it will be possible to cover the whole country.

A Low Cost Transceiver for 19,2 kb/s FSK Transmission in the 23 cm Band

*Wolf-Henning **Rech, N1EOW, DF9IC @ DB0GV.DEU.EU, Pariser** Gasse 2, D-64347 Griesheim, Germany*

1. Why 23 cm ?

Why not ? Yes, it is well known that many packeteers become quite unsure when they see a circuit diagram full of obvious short circuits (some people call them strip lines) connected to parallel capacitors - what a nonsense ! And why to use even discrete transistor stages instead of the fine HCMOS inverters ? Speaking about microwave transmitters finally cause the complete panic, and everyone assures that he has nor a lathe neither a network analyser at hand.

But, don’t panic! 23 cm isn’t yet so exotic as you may think perhaps. A lot of good semiconductors and other components are available at low prices, and with a well-designed printed circuit board the assembly is nearly as easy as for a TNC. Yes, there are some special rules to care, but they are few and simple.

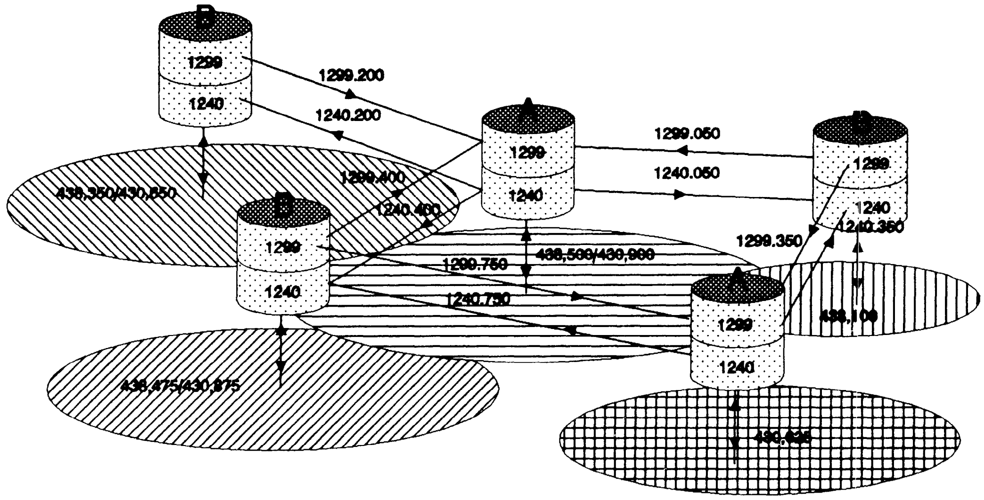


Fig. 1 European network structure: nodes operate with user access in the 70cm band (simplex or repeater like), and several duplex point-to-point links in 23cm

Honestly, this is not the whole truth. The real reason why the author has chosen just 23 cm for his unit is the Central European PR situation where the 23cm band is used for point-to-point-links between the network nodes [1] (see example Fig. 1). A large percentage of these links works with several of the authors former designs (hundreds of units in operation). The latest version of such a simple half-duplex packet radio is presented here, capable to boost your link to 19,2 kb/s with the well known K9NG/G3RUH FSK modem.

2. The concept

To keep the cost low there is only one single microwave frequency generator for both transmitter and receiver. The trick is to use the transmitter frequency as the local oscillator for the receiver, and hence a first IF identical with the Tx/Rx shift. This idea restricts the application to half-duplex radios with enough shift (more than 25 MHz); for the US the subbands 1248-1251 and 1297-1300 with 49 MHz shift are suggested.

The microwave frequency is not generated from an overtone crystal by cascaded multipliers, as usual in transverter circuits. Instead, a simple VCO produces directly the final frequency with very low spurii, without the need for expensive high Q filters and complicated tuning procedures. Its output feeds a frequency divider (a type used in TV sets) with a ratio of 128. The resulting 10 MHz are phase compared to a fundamental crystal oscillator, and the VCO is locked to it. The FM modulation is applied to a varicap in the crystal load circuit - the PLL is so fast that even the 15 kHz FSK modulation is precisely tracked.

The crystal is thermally stabilized by a PTC disk soldered on it - it's a self-regulating system heating up to abt. 110 deg. F when connected to the power supply. Thus the 23cm frequency is stable to 1..2 kHz - good enough for a **30** kHz wide IF filter.

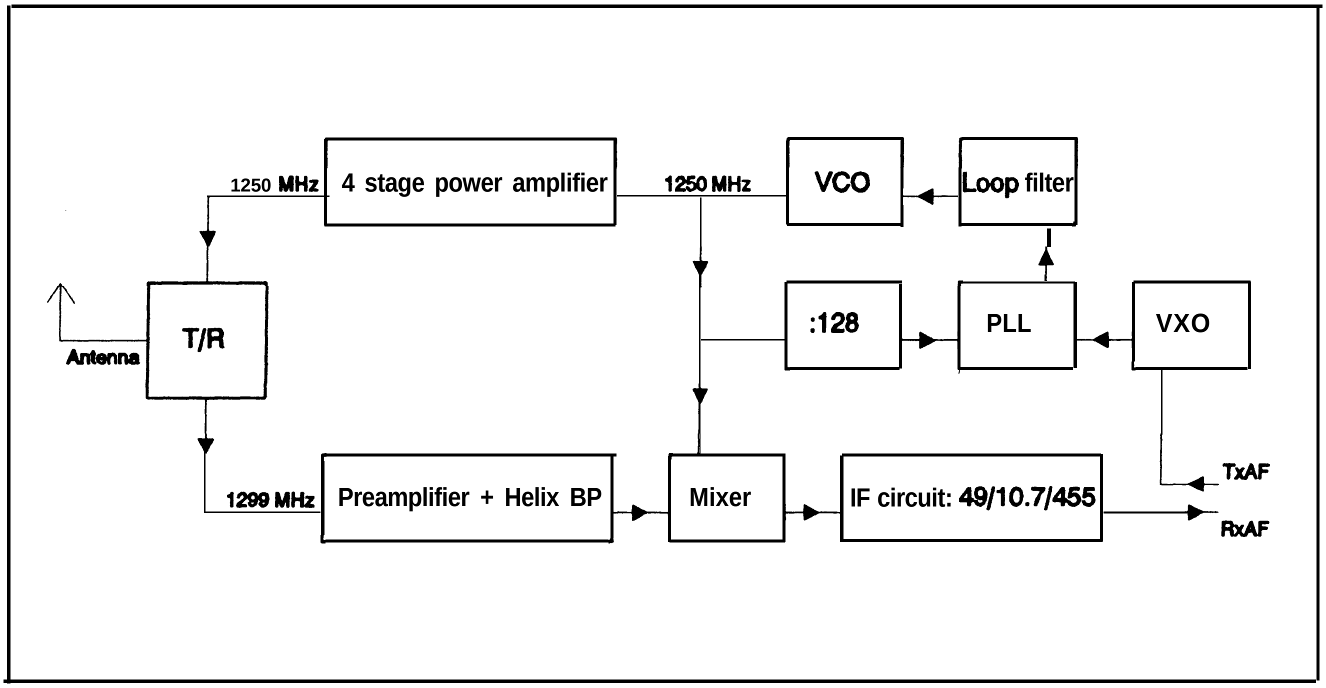


Fig. 2 Block diagram of the transceiver

The **49** MHz IF is downconverted in two steps to 10,7 MHz and 455 kHz, using a crystal and a ceramic filter for adequate channel selection. The 6-pole ceramic filter is a special group-delay-optimized type to achieve the best performance with FSK modulation. A state of the art MC3371 IF amplifier demodulates the signal and provides an additional wide range S meter.

The UHF part consists of a two-stage bipolar preamplifier for an overall noise figure of 3 dB which is a good value as the loss of the T/R switch and the mixer noise is already included. A helix filter feeds the signal to a GaAs dual gate mixer. The transmitter uses low cost Philips bipolar transistors and delivers 0.6 W minimum to the coax. A fast and reliable T/R switch operates with two pin diodes (also TV types).

3. Tuning is simple

What you really need is a 10 Mhz scope and a wattmeter for 23cm (no calibration required, indicator is enough) - careful construction of the unit assumed. The VCO frequency can be tuned by observation of the PLL voltage with the scope - you clearly see when the oscillator locks. The unit produces then a clean signal that can be used to maximize the transmitter gain. and output (Fig. 3).

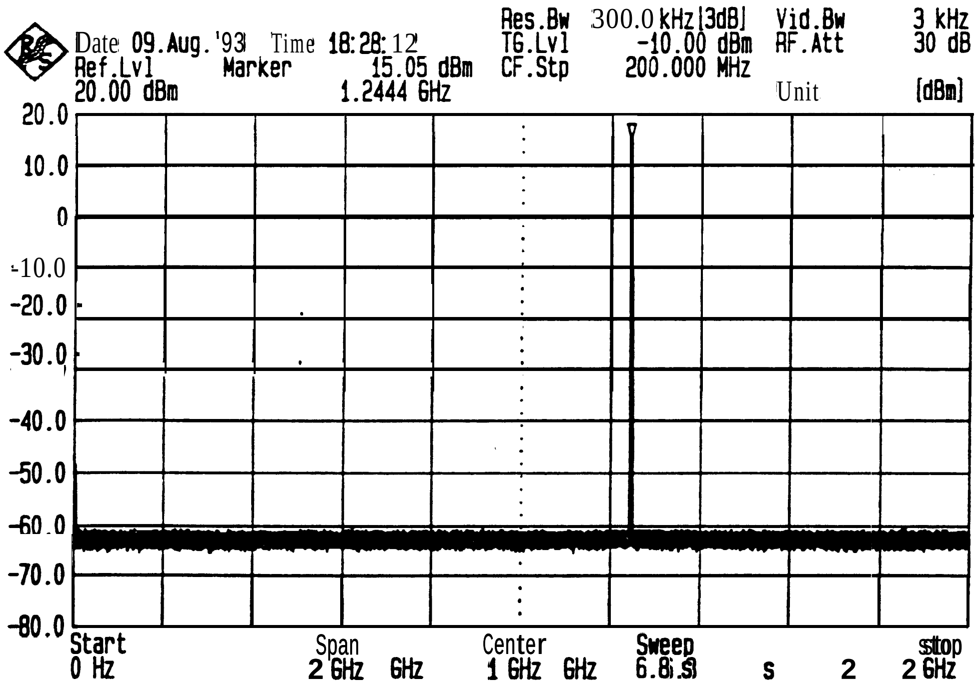


Fig. 3 **Transmitter output spectrum**

For the receiver tuning you need any test signal that is not too strong - e.g. a 70 cm handy transmitting on one third of the frequency. Then tune for maximum S-Meter voltage. The discriminator can be pretuned to maximum audio noise with no signal input, and optimized under operating conditions looking at the eye diagram [2,3].

4. The modem connection

Just 4 wires - transmit and receive AF, PTT, and ground. Use shielded cable for the AF lines to prevent interference from nearby power or digital circuits. The receive level is in the 1 V_{pp} range while the modulator accepts about 2 Vpp. The PTT is switched to ground, with a positive voltage applied, as usual.

The modem itsself has to be modified from the nominal 9k6 design, by doubling the clock frequency, and halfing several capacitor values. Refer to [2] for detailed instructions.

5. How look the signals like ?

Not far from what you know from 9k6 FSK. Both transmitter and receiver are fast enough for a modulation rate of 19.2 kb/s. The sound is as "noisy" as from the 9k6, but the signal cannot be demodulated with common handies or mobile transceivers because its bandwidth is too large for their 15 kHz wide IF filters. During transmission the receiver IF circuits of the unit are not keyed off to minimize the switch-over-time - so audio or noise is present permanently.

6. What about propagation on 23cm ?

Only line of sight ? No, if the distance is not too far some minor obstacles like gentle hills don't matter. But as we use low power we should provide enough antenna gain - look how pretty small even 23cm long yagis are. Or use a dish with 2...6 feet diameter, and an appropriate feed.

Short distances of 20...40 miles non-sight show propagation losses which are usually weather independent within some dBs. In contrast, links over 100 miles between high mountains with line of sight may fail periodically even with a budget of 30 dB above the system margin if inversion layers cross the path.

7. If you want to try such a 23cm link ?

At the time of the manuscript deadline the prototype was just in completion. Kits will be available in Germany at the end of 1993, and also in the US if there is enough interest. Feel free to contact the author for additional information.

8. Bibliography

[1] Rech, W.-H., DF9IC, Kneip, J., DG3RBU: The German (Central European) Packet Radio Network: an Overview. Proceedings 1 lth ARRL Computer Networking Conference 1992.

[2] Miller, J., G3RUH: 9600 baud packet radio modem design. Proceedings of 7th ARRL Computer Networking Conference 1987.

[3] Rech, W.-H., DF9IC: Augen-Diagnostik. ADACOM Magazin 4 (1992), 30-36.

LOW COST ENTRY INTO PACKET RADIO USING BAYCOM
BY CHRISTOPHER C. RENDENNA, **KB2BBW**

This paper picks up where last year's topic, "Low Cost Entry Into Packet Radio Using Digicom" (1 lth ARRL Computer Networking Conference) left off. Its purpose is to acquaint the beginner or experienced packet user with the versatile BayCom Modem as well as providing information on setting up a low cost packet radio system via the BayCom modem and software. It assumes the reader is minimally computer literate and has a basic knowledge and understanding of packet radio basics.

History

BayCom's history began in the Winter of 1989 when Florian Radlherr (DL8MBT), the father of DIGICOM, put his programming expertise and free time to use. The result was the development and completion of a packet terminal program whose operations were heavily dependent upon the software - not the hardware. The first released version was V 1 .OO. Totally in German, it contained the minimal of features as well but a well documented schematic for the BayCom Modem. The next release, version, V1.20, had a three part screen (TX, RX and Monitor) with cursor flexibility to move within each of the three, resizing of the windows, page scrolling, conventional multi-connect ability that included file transfer, EGA video support and screen save feature upon exiting of the program.

Initial response to this version was overwhelming and inspiring. This response, the BayCom Movement as I call it, led to the release of V1.40 in 199 1. Bugs were cleared out, 300 baud HF operation with an AM79 10 or 79 11 chip modem was possible, logbook, disconnect timer, screen clearer, remote control, 50 and 60 line VGA display driver and personalized connect messages were just a few of the enhancements made without sacrifice to previous operational commands. Since this program was still in German, an English version (V1 .40E) was released shortly after.

By the end of 1992, the BayCom group's tireless effort resulted in the completion of V1 SO. Two versions were released: V1 .50 was a European version and V1.50A was a United States version. To protect the hard work of the BayCom Group, licenses were issued to PacComm and Tigertronics to allow commercial distribution of the software in the United States. This action by the BayCom team was in response to those unscrupulous folks who thought they could make a profit by selling the software. The program is copyrighted by Florian Radlherr, DL8MBT and Johannes Kneip, DG3RBU. The English translation and manuals are copyrighted by Denis Godfrey, GOKIU. However, copying of the disk is allowed under certain circumstances. (See Appendix A)

Version 1.50A allows open squelch operation with the DCD command, incorporates a screen saver, definition of standard texts and macros, expansion to support COM3 and COM4, additional cursor movements, disable/enable transmitter, directory display, mouse support, and CW identification (V1.50 English version to conform to law). Those users who own the BayCom USCC-Card (a modem designed on a card for insertion inside of a PC) can enjoy the luxury of, in addition to the TCM3 105 chip, the AM79 11 modem allowing baud rates between 300 and 2400. A G3RUH compatible FSK-modem developed by DF9IC can allow baud rates of up to 9600. Perhaps the most valuable feature is the improvement of the on-line help windows that allow an explanation of every command in the program

Hardware

The original German modem design is based around the TCM3 105 chip which allows both VHF and 300 baud HF operation. BayCom V1.50A will work with the modem design using the 791017911 chip, as utilized by A&A Engineering (See Appendix B and C). PacComm's BayMod-9 (one of many of their models) keeps loyal to the original German design using the TCM3 105 in their modems. Tigertronics, Inc. has a modem that uses a supposedly unique patented chip that draws very little current from the computer making it ideal for laptop use. Additional, there are kits being offered by smaller companies and amateur radio clubs utilizing different schematic configurations. The prospective buyer should always inquire first.

With the exception of the 7910 modem from A&A Engineering which uses an external power source, most BayCom modems are powered off the serial port. The following technical information is courtesy of John - WA6IKO.

“The first thing you need to realize is that there is no “real” source of power on an RS-232 port, so you need to steal the power from the signal lines. Many people glance at the RS-232 specs and conclude you can draw 10ma from the drivers. Wrong! Look closer - 10ma is the “short circuit” current limit from the port. That means if you short the port to ground (zero volts available!) you will be drawing 10ma. The only common CMOS modem chip is the TMS-3 105. Circuits using this chip like to gobble up about 8ma @ 5 volts. To make matters worse, you need to regulate this voltage for the chip to work reliably. There seem to be two approaches to this problem, both with limitations. Many of the circuits floating around use a common 78L05 voltage regulator. The problem here is that the regulator can drop as much as 3 volts across itself in the process of regulating. Add to this a .7 volt loss for the diode that isolates the regulator from the port. Now you need 5 volts (for the chip) + 3 volts (for the regulator) + .7 volts (for the diode) = 8.7 volts @ 8ma. All from this from a port that gives you 10ma if you short it out.

In a “real” BayCom modem, the designers saw the regulator drop problem and decided to use a zener “shunt” regulator circuit. But this has it’s problems also. We still have 5 volts that we need + .7 volts (for the diode) + .95 volts drop through the series resistors that drives the zener (120 ohms @ 8ma) = 6.65 volts. This sounds okay until you realize that the zener is drawing zero current at this supply voltage (which means it’s not regulating!) By the time you find a little more power to get the zener going, your not much better off than you were with the 78L05.

Fortunately, since they knew they were pushing the limits of even the best of RS-232 driver chips, the BayCom folks provided a second source of power in their design. That is why many of the laptop computers (low RS-232 drivers) simply won’t work with the 3 105 circuits.

The BayPac modem by Tigertronics start with special diodes that have only 150 mv drop + a special regulator (not a zener) that only drops 100 mv + a custom chip that only needs 4.5 volts @ 2 ma = 4.75 volts. Compared to the 78L05 design, the BayPac needs only 55% of the voltage and 25% of the current.”

Users should note that earlier designs of the BayPac modem did not use a second source of power.

There seems to be more than several instances of computers re-booting by themselves when using a BayCom modem. This is prevalent to 16Mhz era computers including some the 386-SX computers by AST and PackardBell. Rest assure that this is a computer hardware problem and not a modem or V1.5 software problem. Unfortunately, the only cure is to use J-Comm's "Soft TNC" software. The details of the problem will not be published here, as they point to a company who denies the facts anyway. However, the user should be away of the symptoms and cure.

Ninety-nine percent of all PC compatibles will work with the BayCom Modem. An old PC can be picked up at any hamfest for $100 - $300. Most modem versions utilize a COM port and cost anywhere from $40.00 to $70.00. (See Appendix D) Since the software is available when you buy the modem or from a private source, the cost of a packet set-up becomes quite attractive.

Initial Set-up

The cable configuration will depend on the modem hardware and transceiver. Refer to the company's manual for proper set-up. As for the software, V1 .50A can be run from a disk or hard drive (recommended). Create a subdirectory and copy all the files to it. Some disks will have an install program which will make this easier. There are basically twelve files in V1.50A:

1. L2.EXE This is a memory resident program which allows BayCom to send and receive packets after one exits the program.
2. SCC.EXE User interface program.
3. OFF.COM Removes L2 from RAM.
4. SCCINI ASCII file containing user default parameters
5. PARA.EXE Converts SCCINI ASCII file into machine language.
6. SCC.PAR File created by PARA.EXE from SCCINI.
7. TERMHELP.SCC BayCom help file.
8. SCC.VID If the feature is activated, BayCom will save the contents of the three split screens upon exiting.
9. SCC.PWD Password file that must be created if the :J command is to be used.
10. BAYCOM.BAT Run this batch file to start BAYCOM.
11. INSTALL.BAT Batch file that will create a BayCom subdirectory and then copy the files to it.
12. SCC.LOG If the :LOG command is on, the program will create this file and write to it a list of all the stations that were connected.

Before one runs the program, a few changes should be made to the SCCINI file. Use you favorite word processor, open the file and edit the COM port setting to the appropriate number. Enter your call sign and SSID under the MY parameter, which should be different from the SSID under the digipeater (DCALL) parameter. Personal preferences will determine the content you place in the connect text (CTEXT) and quit text (QTEXT) fields. Note that the semi-colon (;) at the beginning of each line acts like a REM statement. The line will be ignored.

Getting on the Air

After executing the batch file "BAYCOM", the program will run and three screens will appear. The top is the transmit, middle is the receive and the bottom is the monitor screen. The user should be sensitive to the fact that all commands are entered after the colon (:), while anything typed without the colon and then followed by a carriage return will automatically be transmitted over the air. Die-hard Digicom users will recognize this difference immediately. :HBAUD should be set at 1200, FRACK around 5-10, TXDELAY around 40-50, then fine tuned down to around 30 or 40 depending on the transceiver type. Electronic switching relays do not need as long of a TXDELAY value as do the older mechanical relays. The DWAIT parameter is relative to the amount of traffic on the frequency and should be set at a level that is courteous to all. The on-line help or manufacturer's manual should be reviewed before hand for more in-depth information and additional features.

Tricks of the Trade

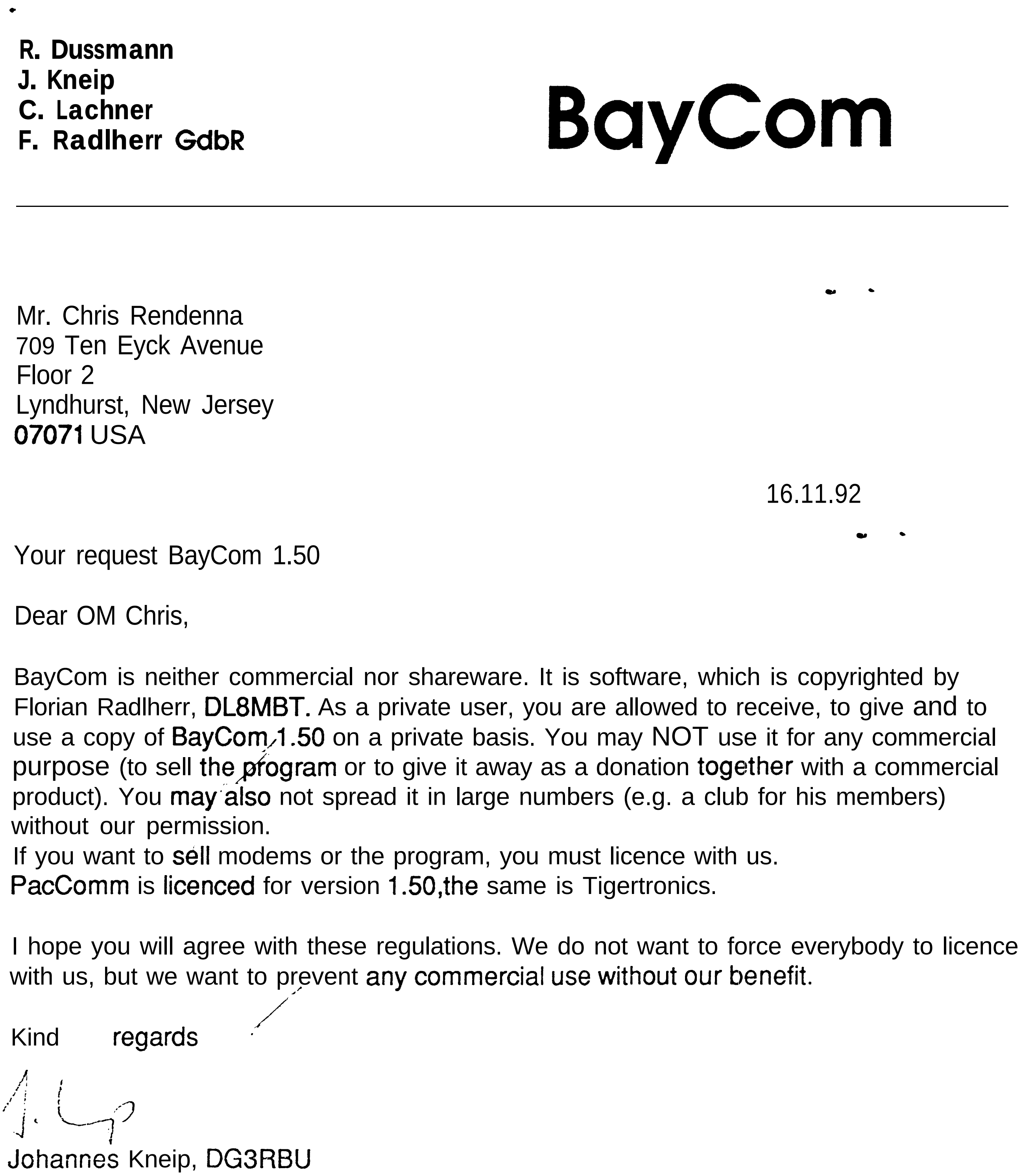
BayCom is a versatile terminal program. It was never meant to be a mailbox, yet I receive numerous requests for information on how this can be done. Well, when there is a will, there is a way! Just as in V1.4, the user can specify which commands would be allowed to be used for remote. They are entered in the RCMD line of the SCC.INI file. Set the REMOTE parameter to on. By leaving the computer and transceiver on continuously, one can simulate a mailbox. Simply edit the CTEXT announcing to connected stations that they may leave a message using the //W <filename> command. (All remote commands sent by another station follow two slashes). File names should be limited to eight characters. After text is sent by the connecting station, the command //W OFF, is sent and closes the file. Mail can then be viewed with the :V <filename> command by the host.

For the more daring, one can insert the OSHELL command in the RCMD line. Users will be able to access your computer (and hard drive) by using the command //OSHELL just as if it was their own computer. This setting is dangerous as an evil minded user could actually delete your whole hard drive! If you are determined to set up your system like this, have the common sense to change the attributes of important files to read only or hidden. There is mini-bbs program called Multi-User BayCom which is recommended for the mailbox-minded user. Sources and software descriptions are contained in Appendix D and E respectively.

The simplicity of the BayCom modem makes it compatible with other software packages as well. (See Appendix E) To cover each and all of these would go beyond the scope of this paper. One should keep abreast of the latest developments by acquiring a copy of the Original BayCom Exchange. (See Appendix D). BayCom can be an inexpensive way to open up into the world of packet radio. Try it!

[Chris Rendenna, KB2BB W/AAR2MG, caught the packet bug from his elmer Jack, K2ZBR back in 1983. Chris then got involved with TCP/IP through his father, Vince, N2CLR, who is a Digitial Networking Specialist at AT&T Discovering Digicom in 1985 led Chris into the world of BayCom. Finding no literature on Digicom or BayCom, Chris created the Original Digicom Exchange in 1990 and the Original BayCom Exchange in 1993. These publications, voluntarily put together, were the first and still are the only regular means of bringing together Digicom and BayCom users alike. Currently, the Exchanges have unexpectedly reached six continents. Chris hopes the Exchanges might make WAC some day - hi!]

APPENDIX A



R. Dussmann
J. Kneip
C. Lachner
F. Radlherr GdbR

BayCom

Mr. Chris Rendenna
709 Ten Eyck Avenue
Floor 2
Lyndhurst, New Jersey
07071 USA

16.11.92

Your request BayCom 1.50

Dear OM Chris,

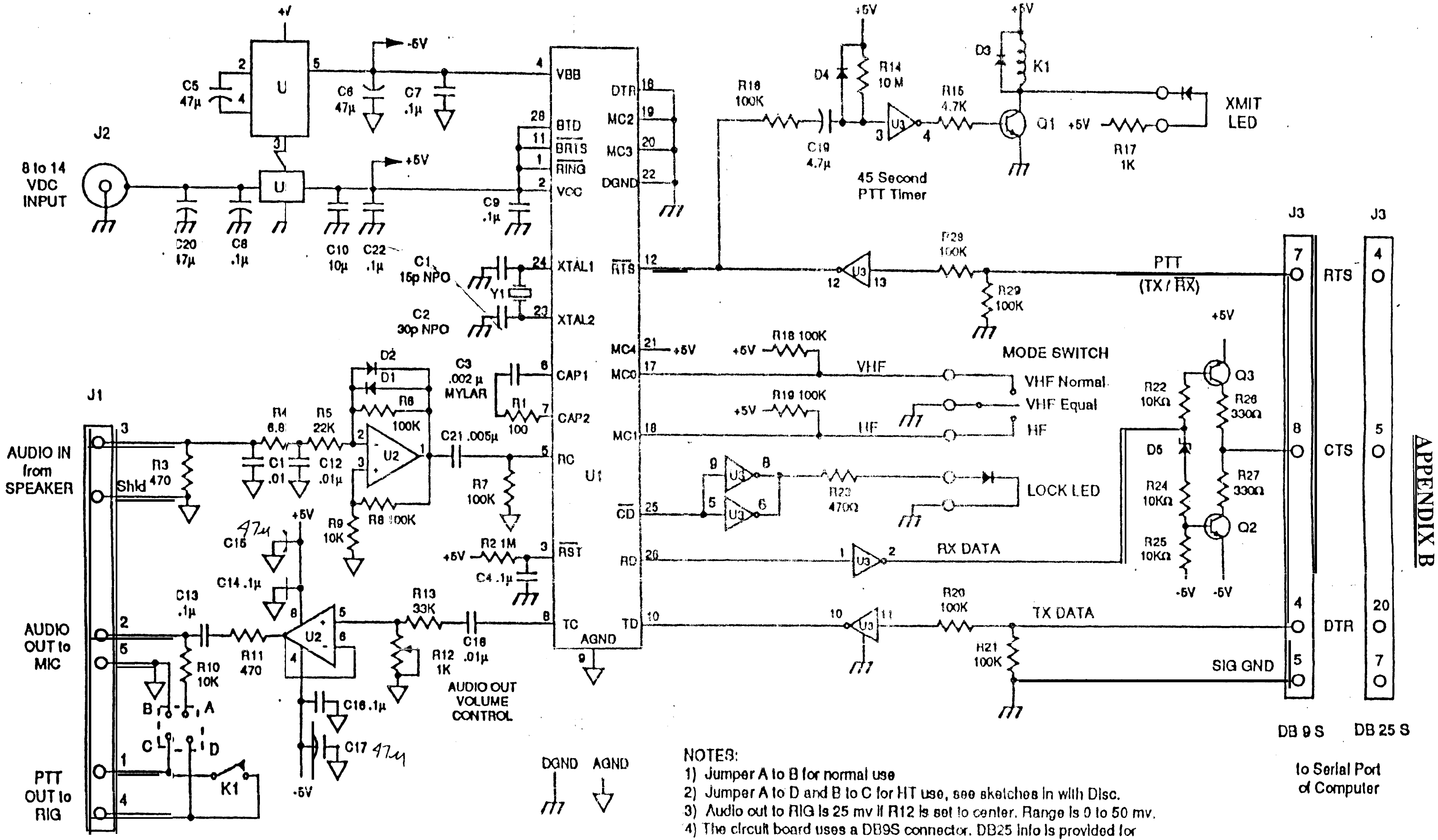
BayCom is neither commercial nor shareware. It is software, which is copyrighted by Florian Radlherr, DL8MBT. As a private user, you are allowed to receive, to give and to use a copy of BayCom 1.50 on a private basis. You may NOT use it for any commercial purpose (to sell the program or to give it away as a donation together with a commercial product). You may also not spread it in large numbers (e.g. a club for his members) without our permission.
If you want to sell modems or the program, you must licence with us.
PacComm is licenced for version 1.50,the same is Tigertronics.

I hope you will agree with these regulations. We do not want to force everybody to licence with us, but we want to prevent any commercial use without our benefit.

Kind regards

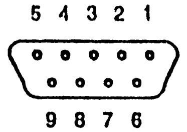
Johannes Kneip, DG3RBU

APPENDIX B

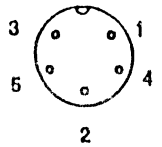


NOTES:
1) Jumper A to B for normal use
2) Jumper A to D and B to C for HT use, see sketches in with Disc.
3) Audio out to RIG is 25 mv if R12 is set to center. Range is 0 to 50 mv.
4) The circuit board uses a DB9S connector. DB25 info is provided for those who wish to make an adapter

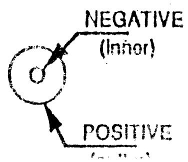
Mating 9 pin D
as viewed from the wiring side



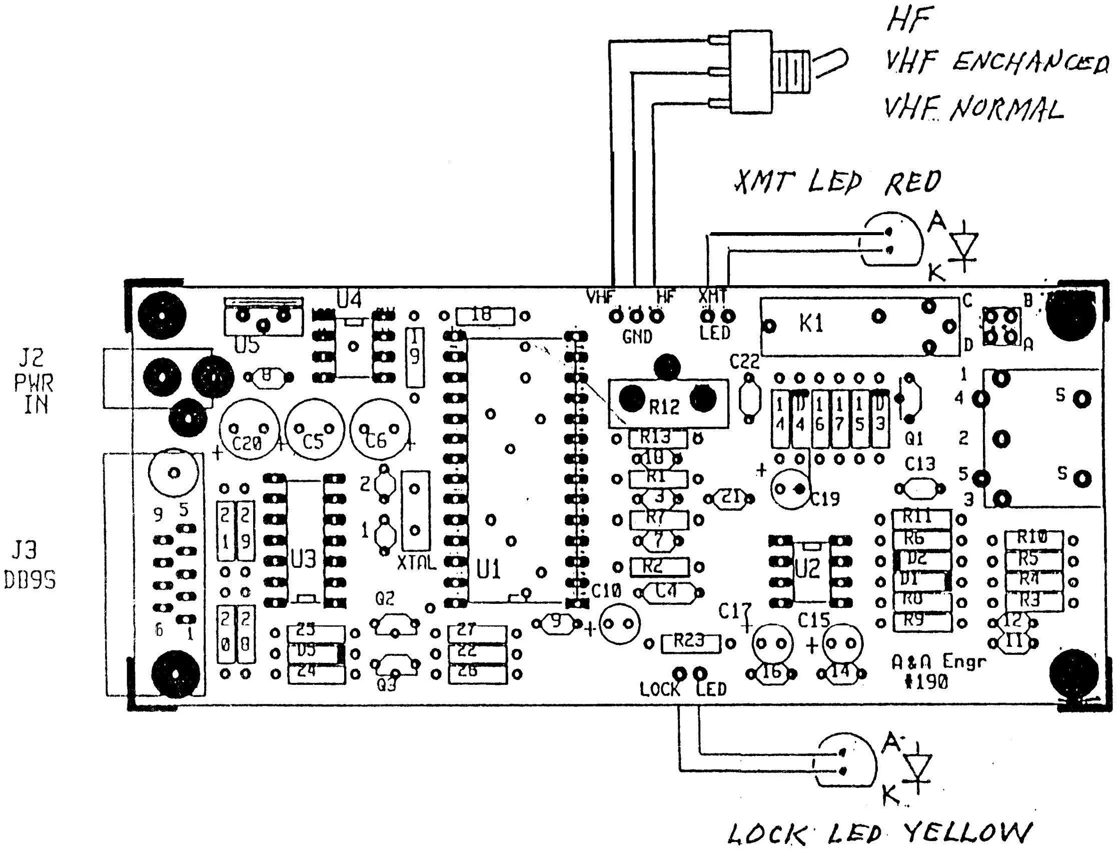
Mating 5 pin DIN
as viewed from the wiring side



Mating 2.1 mm Power Plug
as viewed from the wiring side



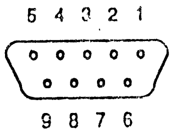
A & A Engineering	2521 W. La Palma, Unit K Anaheim, CA 92801 USA	
SCALE n/a DATE 26 APL 91	de W6UCM	DRN BY STAS A REV 13 MAY 92
Baycom Modem		
Schematic	DRAWING NUMBER 492-190	REV 0



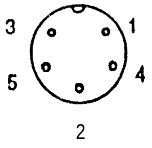
REF DES	QTY	DESCRIPTION
R1	1	100 n 1/4 W 5% BN-BK-BN
R2	1	1 MΩ 1/4 W 5% DN-BK-GN'
R3,11,23	3	470 n 1/4 W 5% YL-VI-EN
R4	1	6.8 KΩ 1/4 W 5% BL-GY-RD
R5	1	22 KΩ 1/4 W 5% RD-RD-OR
R6,7,8,16 18,19,20 21,28,29	10	100 KΩ 1/4 W 5% BN-BK-YL
R9,10,22 24,25	5	10 KΩ 1/4 W 5% BN-BK-OR
R12	1	1 KΩ 1 Turn Vatt Pot
813	1	33 xn 1/4 W 5% OR-OR-OR
R14	1	10 MΩ 1/4 W 5% BN-BK-BL
R15	1	4.7 KΩ 1/4 W 5% YL-VI-RD
R17	1	1 Kn 1/4 W 5% BN-OK-RD
R26,27	2	330 Ω 1/4 W 5% OR-OR-DH
Cl	1	15 pF NPO Disc
C2	1	30 pF NPO Disc
c3	1	.002 µF Mylar (202 or 222)
C4	1	.1 µF Mylar (104)
C5,6,20	3	47 µF Radial
C7,8,9,13 14,16,22	7	.1 µF Disc/Mono (104)
Cl0	1	10 µF Radial
C15,17	2	47 µF Radial
C11,12,18	3	.01 µF Disc/Mono (103)
C19	1	4.7 µF Radial
C21	1	.005 µF Mylar (502)
D1,2	2	1N60/270 Ge Diode (or equivalent)
D3,4	2	1N4148 Si Diode
D5	1	1N5231B 5.1 V Zener Diode
Q1,2	2	PN2222A Transistor
03	1	PN2907A Transistor
U1	1	AM7910 xc
U2	1	LM1458 IC
u3	1	CD4584 IC or CD40106
U4	1	MAX660 x or LT1054 CN8
US	1	7805 PC
SOC	2	8 pin x c socket
SOC	1	14 pin IC socket
SOC	1	28 pin IC socket
K1	1	5V Read Relay
Y1	1	2.4576 Mhz Crystal
J1 (card)	1	3 pin DIN PCB Connector Radio x/o
31 (cable)	1	5 pin DIN Cable Connector Radio I/O
32	1	2.1mm PCB Power Connector
J3	1	9 pin D PCB Signal Conn Serial I/O
SW1	1	SP3T toggle switch
LCD	1	RED LED w/ mntg clip (XMT)
LED	1	YEL LED w/ mntg clip (LOCK)
HRDW	1	4 Pin ,100 Header
HRDW	2	Shorting Bars
PCB	1	#492-190 Circuit Board

APPENDIX C

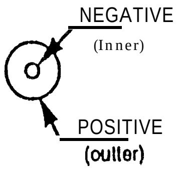
Mating 9 pin D
as viewed from the wiring side



Mating 5 pin DIN
as viewed from the wiring side



Mating 2.1 mm Power Plug
as viewed from the wiring side



A & A Engineering		2521 W. La Palma, Unit K Anaheim, CA 92801 USA
SCALE n/a DATE 26 APL 91	de W6UCM	DRN BY STAS A REV 13MAY92
Baycom Modem		
Component Layout		DRAWING NUMBER REV 492490 I 0

APPENDIX D

SOURCES FOR HARDWARE

A&A Engineering, 2521 LaPalma, Unit K, Anaheim, CA 92801 (714) 952-2114
(714) 952-3280 FAX
Crawford Amateur Radio Society, PO Box 653, Meadville, PA 16335
PacComm Packet Radio Systems, Inc., 4413 N. Hesperides Street, Tampa, FL 33614-7618
(813) 874-2980 (813) 872-8696 FAX
Ramsey Electronics, Inc., 793 Canning Parkway, Victor, NY 14564 (716) 924-4560
Tigertronics, Inc., 400 Daily Lane, PO Box 5210, Grants Pass, OR 97527 (503) 474-6700
(503) 474-6703 FAX

SOURCES FOR SOFTWARE

(PacComm and Tigertronics are currently the only two companies licensed to sell BayCom V1 .50A with their modems, however, amateurs may obtain a copy of the software from the address below as long as they comply with the conditions as stated by the BayCom Team in Appendix A).

BayCom V1.5, Multi-User BayCom, and others may be obtained by sending a SASMailer + floppy (360,720, or 1.44) to:

Chris Rendenna, KB2BBW
709 Ten Eyck Avenue
Lyndhurst, NJ 07071 USA

SOURCE FOR BAYCOM PUBLICATIONS

Currently, there is only one source of current information regarding BayCom. Send a SASE for the latest issue.

The Original BAYCOM EXCHANGE, Chris Rendenna, KB2BBW, Editor
709 Ten Eyck Avenue, Lyndhurst, NJ 07071
KB2BBW @ WA2SNA.NJ.USA.NA

ACKNOWLEDGMENTS

Many thanks to Stas W6UCM of A&A Engineering for permission to print his BayCom 7910 chip modem design.

I would also like to take this opportunity to thank all the BayCom users throughout the world who have assisted me in various ways with this paper. Most of all, I want to thank the BayCom Team in Germany for producing a fine and enjoyable product.

APPENDIX E

***** SOFTWARE GUIDE FOR THE BAYCOM MODEM *****

The following guide was first published in the February 1993 issue of the Original BAYCOM EXCHANGE (Vol. 1, No. 2). Many thanks to Jeff (N9NGF @ W9QYQ.IN), Denis (GOKIU) and G6IIM who helped with the list and sources.

BAYCOM V1.2 Terminal Program for PC's without a TNC by Florian Radlher (DL8MBT) and Johannes Kneip (DG3RBU). Developed in 1989/90. Screen editor, 3-part split screen (TX, RX and Monitor), scroll capability, 8 ports, supports EGA video, screen save. The only bug is an inoperational DIG1 feature. (See Vol. 2 No. 12 December 1991 of DIGICOM EXCHANGE)

BAYCOM Vl.3 This is a little know version that is completely in German. Similar to V1 .2 with a few enhancements.

BAYCOM V1.4 New updates to V1 .3 includes: Operation with AM79 10/79 11 modems, system messages are switchable between German and English, log file, DISC timer, switchable between Insert and Overwrite, DOS Shell, remote control within limits, 50 and 60 line VGA display, Personalized connect text, and more. Although DIG1 bug has been fixed, BTEXT and QTEXT now have bugs. (See Vol. 2 No. 12 December 1991 of DIGICOM EXCHANGE)

BAYCOM V1.5 Supports COM 1 and COM 2. Bugs seen in VI.2 and V1.4 are fixed, but now there seems to be a keyboard buffer bug in the original V1 .5 experimental version. Later official prove to be bug-free. This version for European distribution.

BAYCOM **V1.50A** This is the copyrighted US version of BayCom. All bugs have been corrected, additional ports are supported. More parameters in SCC.INI.

ESKAY PACKET Also known as SP or Super Packet by DL1MEN and DL1BHO. The last public domain version was V6.11. V7.00 is no longer public domain. The latest version is V7.50, which contains and English manual The program has a similar screen layout as BayCom, but has many more features, such as 35+ remote commands, autorouting, binary transfers, and up to 10 simultaneous connections, to name a few. In order to use it with the BayCom modem. Must use the driver called TFPCX.EXE. This driver is what actually does the work of the TNC.

GRAPHIC PACKET Written by Ulf Saran (DH1DAE) Latest public domain version is V1.52, released just this summer. Very impressive program that works with the TFPCX.EXE driver available in Eskay Packet. It does not have as many features as Eskay Packet, but is easy on the eyes. It has auto binary transfers, on-screen clock, up to 10 multi-connects, and mouse support.

TFPCX (T)he (F)irmware (PC) E(x)tended) Resident AX.25Controller for PC without TNC by Rene Stange (DGOFT). This is a driver program that allows BayCom modems and USCC Cards to use SP and GP. TFPCX is compatible with The Firmware from NORD><LINK and runs resident in the background as an AX.25L2-Controller on IBM Compatible PC's (Not on ATARI ST). TFPCX V2.3B now has a Soft-DCD (Programmable Squelch Barrier) and makes possible an internal self connection for internal Test work. It will not work well with'computer speeds under 8 Mhz.

	PC	XT	XT	286	386
Baud	MHz	5	8	12	20
300		*	*	*	*
1200		?	*	*	*
2400		/	?	*	*
4800		/	/	?	*

* - Operation possible
? - Operation possibly with some restrictions
/ - Operation not possible

WG7J NOS Note that there are many different versions of NOS. You can find NOS on just about any good ham bbs. You will also need a driver to use the BayCom modem. The most current one at this time is listed below:

AX921123 Some of the earlier versions of this driver had bugs, so beware. File name denotes release (November 23, 1992) and most likely there is a more recent version floating around on the land-line ham BBS'.

Improved TNC Interconnections

Donald A. Rotolo, N2IRZ
Amateur Networking Supply
PO Box 219, Montvale NJ 07675

Abstract:

Hardware for interconnecting TNCs to form a TheNET or ROSE network hub has been further refined for greater convenience. An updated version of the well-known Diode Matrix Board for RS-232 network hub backbones is introduced. The new board has been re-packaged into a backplane-line configuration, eliminating the need and expense of ***cables for the RS-232 signals. Additionally, a new circuit is described which allows multiple TNCs to be interconnected at high speed via their modem disconnect headers, with full flow control, using only two signal wires.***

For effective Packet Radio networks, it is essential that a three-dimensional network topology be utilized. This means that user access and data transport must be not performed on the same radio channel. Generally, this is done by having a network user access channel on the 2 meter band and a second network TNC sending the data into the network on a backbone channel using a different band, such as 440 MHz. In small or lightly loaded networks, a number of 2-port network hubs may be effective. However, as the data loading increases, it becomes necessary to rely upon point-to-point links. This creates a need for network switching hubs of three or more TNCs, which must be interconnected on a Local Area Network, or Matrix. Traditionally, the lowest cost implementation for such a matrix has been the Diode Matrix Board.

The Diode Matrix Board was originally produced by John Painter, N0NDO, in 1989 as the TJP Octopus. It had eight ports, and the wires to the TNC connectors were soldered directly to the PC Board. The Octopus, with its twisted maze of cables emanating from the tiny PC board, was the best solution available at that time.

In early 1991, the North East Digital Association' began selling the NEDA Hexiyus, a six-port version of the Octopus. The Hexipus was built mostly for NEDA's use in their growing TheNET network, since Octopus boards were no longer available and hand-made matrices built on perfboard were not consistently reliable. The redesigned PC Board addressed two of the Octopus' biggest problems: the soldered-in cables and RS-

232 fan-out problems with some TNCs. The Hexipus used 9-pin sub-D connectors, allowing the use of commonly available, removable cables. Limiting the number of ports to six resolved the fan-out situation (where some TNCs could not provide sufficient RS-232 drive current, stopping communications), as well as reducing the backbone contention at busy sites (where there is more data to be handled than time in which to handle it, causing slowdowns and data collisions). The Hexipus is no longer being produced.

At approximately the same time, the ROSE networking software was upgraded to support diode matrix interconnection, unfortunately using a different pin-out from the Hexipus, which had been designed with TheNET in mind. Previously, ROSE networks had to use the relatively expensive Active RS-232 LAN designed by Tom Moulton, W2VY. Bill Slack, NX2P, designed the five-port EZ-Matrix[2], mostly because he needed an inexpensive, easy to build circuit to implement a ROSE network in northwestern New Jersey. The EZ-Matrix, which used "HPFM Technology", was the first to be compatible with both ROSE and TheNET networks. It used DIP diode arrays, as opposed to the 60 discrete diodes in the Hexipus, allowing it to be assembled in a few minutes. The networking software type was selected by soldering the DE-9 connectors into different sets of holes in the PC Board.

Recently, the diode matrix board concept has been further refined, as the NETRIX Diode Matrix Board[3]. This six-port circuit was designed with a backplane-like PC board, allowing the TNCs to be connected directlv to the matrix., eliminating cables. The Netrix is compatible with both TheNET and ROSE networking software, which is selected by DIP shorting jumpers. The extremely short data path adds less capacitance to the RS-232 lines, permitting higher speeds. The unique stacking method, with all of the TNCs placed on their side, contributes to better cooling airflow, allowing TNCs to run cooler and therefore more reliably. An added benefit to the elimination of the cables is a neat, professional appearance for the network hub. The Netrix is endorsed by NEDA, RATS[4] and NAPRA[5], leading packet networking organizations.

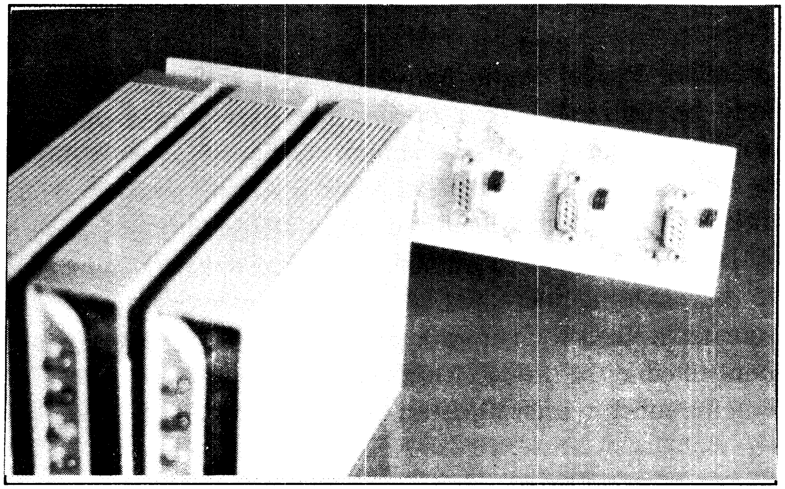


Photo 1: The NETRIX Diode Matrix Board, shown with three TNCs attached.

The Netrix is offered with 9-pin female sub-D connectors, spaced on 1.625 inch centers, allowing any brand of TNC-2 compatible to be used. Commonly available 9 to 25 pin adapters are used to attach TNCs with 25-pin RS-232 connectors, while a 9-pin gender changer is used to connect TNCs with female DE-9 connectors. TNCs with male DE-9 connectors attach directly to the Netrix board.

The Netrix, like the original TJP Octopus, consists of a matrix of diodes, connected such that signals may flow in only one direction. TNCs are designed to be connected to a computer using a straight-through RS-232 cable (TXD to TXD, RXD to RXD, etc.), like most telephone modems, making it a DCE (Data Communications Equipment) device. DCE devices send data on the RXD pin, and receive data on the TXD pin - just the opposite of what you'd expect. The opposite of a DCE device is a DTE (Data Terminal Equipment) device - a computer or terminal - which sends data on the TXD pin and receives data on the RXD pin.

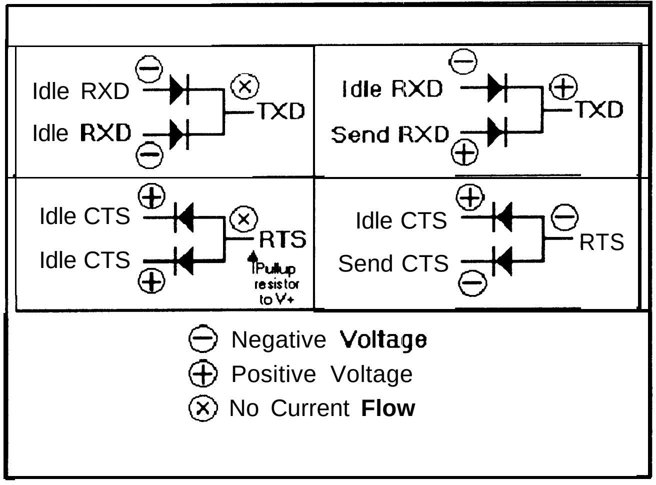


Figure 1: The simplified 2-port diode matrix.

To connect same-type (DTE or DCE) devices together, you use a "null-modem" cable, where all pins on one end are connected to their opposites on the other end to the cable - e.g., TXD is connected to RXD, and so on. A Diode Matrix is therefore wired somewhat like a null-modem, except with diodes having reversed polarity, making the diode matrix a "null-terminal". This permits more than two devices to share the data and control lines.

Diode matrix operation is easiest to explain in terms of idle states and voltage levels. The purpose of the diodes is to allow ANY active (non-idle) line to override ALL idle lines. Some important things to understand:

1) RS-232 signals may have either of two voltage levels, + 12 volts and - 12 volts.
2) Data signals go through the diodes in the opposite direction from the Flow control signals.
3) Current flows through a diode only when the voltage at the cathode (the side with the line or band) is lower than the anode voltage.

For DATA lines:

Idle = Mark = 1 = V- **Data = Space = 0 = V+** **One V+ must override many V-**

For FLOW lines:

Idle = Space = 0 = V+ **Flow = Mark = 1 = V-** **One V- must override many V+**

So, to cover all four possible cases (all Data idle, all Flow idle, one Data active, one Flow active), the simple 2-port diode matrix must look like the circuit shown in Fig. 1.

Occasionally, two or more co-located TNCs must be connected via their modem ports. This was often done at audio levels, using a null radio cable. This method is limited by the modem in the TNC, usually 1200 baud. Two TNCs could be connected via the modem disconnect header signals, but adding a third TNC created significant problems.

A simple, inexpensive circuit, designed by Bill Slack, NX2P, is now available to connect six or more TNCs together via their modem disconnect headers. This circuit, dubbed a WireModem Adapter?, operates independently of the type of software running on the TNC and uses only two wires for data and flow control. It is easily (and reversibly) installed into the TNC, and the TNC remains removable from the network hub with the WireModem installed.

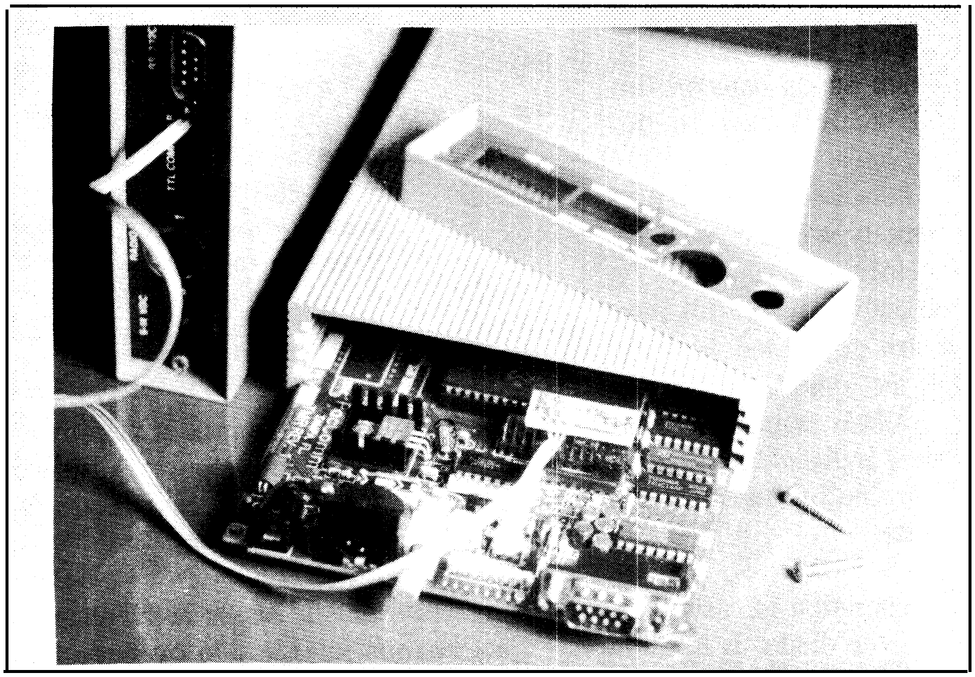


Photo 2: The WireModem Adapter, shown installed in a TNC.

Applications for the WireModem include the creation of a gateway between ROSE and TheNET networks; connection of two diode matrices together for a site with more than six radio ports, allowing high-speed connectivity while maintaining isolation to avoid matrix contention; the addition of a TNC- or computer-based data server (e.g., BBS, Weather Server, DX Cluster, Crowd Node, Callbook Server, etc.); and allowing an easy method of monitoring site activity, as well as site testing and maintenance.

The WireModem adapter is available in kit form, which consists of a small (0.5 x 1.1 inch) PC board, components and connectors. A DIP header is fashioned into a simple interconnecting matrix, permitting a TNC to be easily disconnected for service or inspection. The PC board mounts directly onto the 20-pin modem disconnect header, and the signal and ground wires of each WireModem are all connected in parallel.

The function of the circuit is as follows:

FLOW CONTROL: When the transmitting TNC asserts RTS (Request To Send output, Pin 5, active low), the diode conducts, and the signal is imposed upon CD (Carrier Detect input, Pin 1, active low). The signal is also sent, via the FLOW wire, to the receiving TNCs. In the receiving TNCs, CD is asserted, but the diode blocks the signal from reaching RTS. Thus, the transmitting TNC lights both the PTT (RTS) and DCD (CD) indicators, while in the receiving TNCs only DCD is indicated.

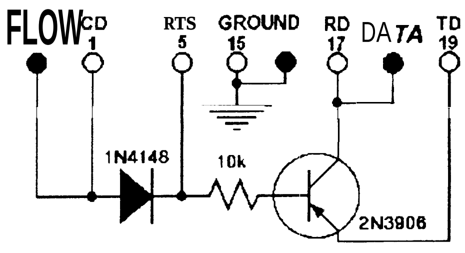


Figure 2: Schematic of the WireModem Adapter.

DATA: The RD input (Receive Data, pin 17) is directly connected to RD in the other TNCs. If RTS is inactive (high), it biases the PNP transistor off, effectively cutting TD

When the transmitting TNC asserts RTS, the transistor is biased on through the resistor, connecting TD to RD. The RD of the

TNCs and

transmitting TNC also hears its own data.

In conclusion, the NETRIX is an updated version of the familiar diode matrix board,

use. The WireModem Adapter simplifies a once difficult task, offering a new solution to a number of networking problems.

Endnotes:

1. PO Box 563, Manchester NH 03 105. NEDA operates a TheNET network spanning from Maine to

2. Available from NX2P Electronics, 321 East Shore Trail, Lake Mohawk NJ 07871

3. Available from Amateur Networking Supply, PO Box 219, Montvale NJ 07645.

4. The Radio Amateur Telecommunications Society, PO Box 93, Park Ridge NJ 07656. RATS is the distributor of the ROSE X.25 Packet Switch networking software, as well as other packet-related software.

5. The Northwest Amateur Packet radio Association, PO Box 70405, Bellevue WA 98007. NAPRA operates a TheNET network in the northwestern United States.

6. Available from Amateur networking Supply, address above.

Packet Tracker
A Graphical Packet Tracking Program

Mark Sproul, KB2ICI
Radio Amateur Telecommunications Society

Mark Sproul, KB2ICI
1368 Noah Road
North Brunswick, NJ 08902
AppleLink: Sproul.M
Internet: sproul@sproul.com
Packet: KB2ICI@KB2EAR.NJ.USA

ABSTRACT

Packet Radio is a very interesting and growing part of Amateur Radio. For the new person investigating Packet Radio, there are many things to be learned and lots of different aspects of Packet Radio that will not be grasped for a long time. Even the advanced Packeteer usually does not understand everything going on with Packet Radio. If someone wants to know what kind of Packet services and what BBS's or other activity is going on in their area, they are usually told to hook up their TNC and put it into MONITOR mode to see what is going on. This process will give them screen after screen, day after day, of almost unintelligible garbage flying across their screen. Unless they take the time to start writing down call signs and understand some of the different protocols, they will be COMPLETELY lost and confused.

Packet Tracker is a program that monitors that stream of 'garbage' and presents a graphical representation of all of the different conversations occurring on the frequency at one time. It does this by showing icons representing the different stations and draws lines from one station / icon to another representing who is talking to whom. It also shows where retries are happening and who was the most recent person to actually transmit and how many packets were transmitted by each station. **Packet Tracker** also keeps track of lots of statistics about the activity of each station and also statistics about actual band usage.

INTRODUCTION

Packet Tracker is the first program in a series of Packet Utility programs specifically designed to teach us something about what is going on in our complex hobby. **Packet Tracker** 'TRACKS' packets and displays what station they are coming from and what station they are going to. **Mail Tracker** is the second in this series, it 'TRACKS' entire mail messages. See the article on Mail Tracker elsewhere in these proceedings.)

Packet Tracker keeps statistics about each station it sees and also statistics about band utilization and retries. This program shows all of this information via graphical representation on the computer screen. It displays stations as ICONs, it shows different types of stations as different ICONs, it show which station transmitted most recently and how much information each station transmits. It also shows who is talking to who by drawing lines between the different ICONs. In the case of multi user BBS's and digipeaters, a single ICON/Station may have several different lines into and out of it. In the case of a DX Cluster, there may be many lines going to many different stations. **Packet Tracker** also keeps track of retries, band-width utilization and other statistics.

DISTRIBUTION

Packet Tracker is freeware. I encourage you to give this program to as many packet users that have Macs as you can. I would, however ask that you send me a QSL card stating that you are using my program and what type of Macintosh you are using it on. Copies of the program will be made available on the Internet and on various on-line services such as CompuServe and America On Line. It is also available on the April 1993 Buckmaster's CD-ROM. You are welcome to put a copy of the program on any BBS's provided that the entire package and documentation are included.

OPERATION

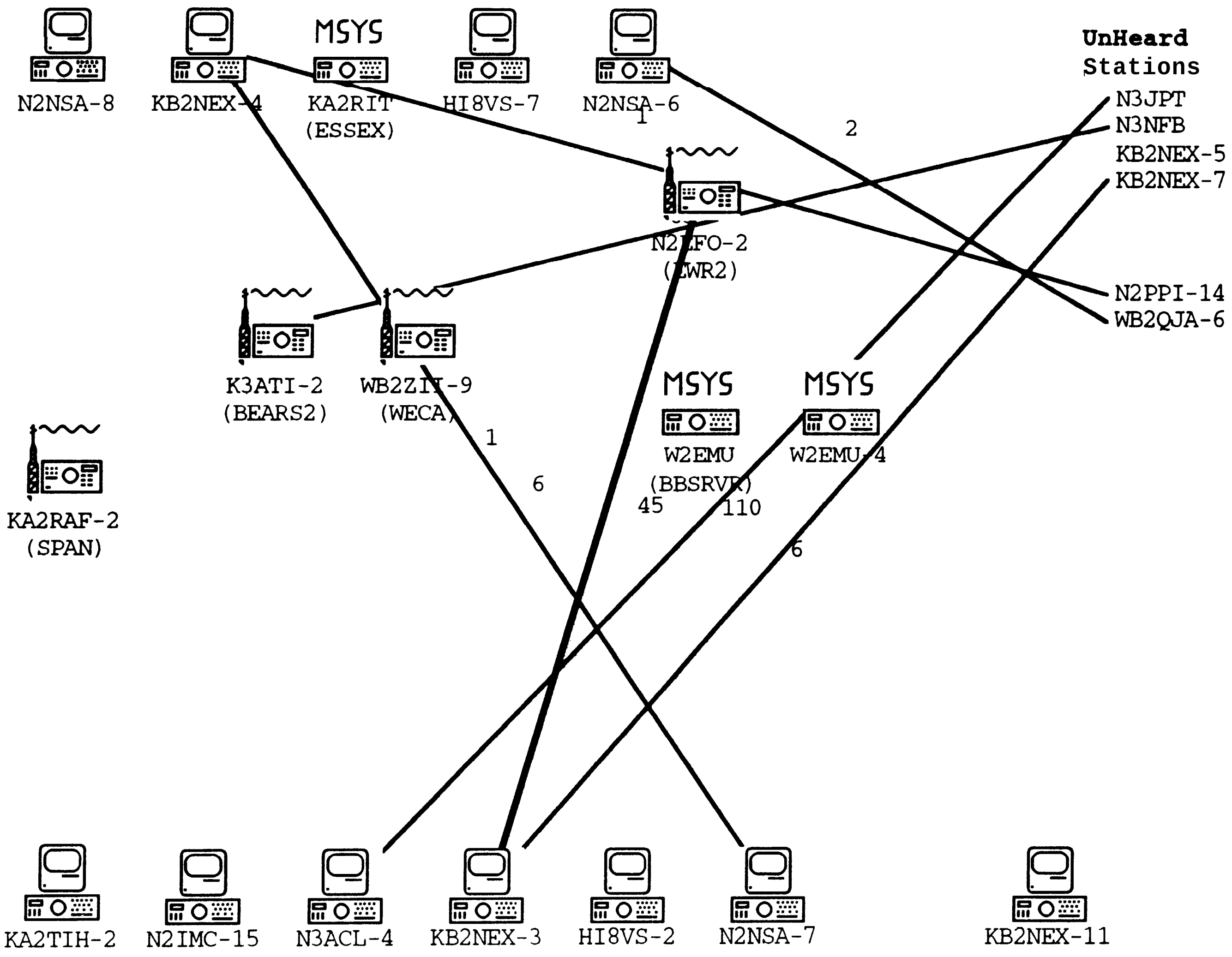
Packet Tracker runs on any Macintosh with System 6.0.5 or later (System 7.0 preferred) and needs the Comm Tool Box installed. (This is standard with System 7.0). It will work with any TNC that has a MONITOR mode. **Packet Tracker** has been tested with AEA, PACCOM, Kantronics and MFJ. It will not work with a KISS only TNC. **Packet Tracker** monitors all of the information that is received from the TNC and tries to make sense of it. It keeps lots of internal data structures for keeping track of all of this information.

There are four windows in Packet Tracker. The main window is **the Station Map Window.** The other three windows are the **Station List Window**, the **Bar Chart Window,** and the **Communications Window. You** can also get information about a specific station by double clicking on it which brings up a fifth window which is the **Information Window.**

Packet Tracker can optionally be set up to do long term statistics. This statistics get written to a log file every hour. This is useful if you want to find out the overall channel usage or other types of statistics about Packet usage.

Station Map Window

The most important window in this program is the map window. It shows all of the stations and data paths graphically. There are two types of stations displayed on the map, heard stations and unheard station. A station that we can hear directly is a heard station. It is displayed as an icon on the map. The default icon is a radio with a computer sitting on top of it. If the program can make any determination as to what the station is, it will automatically change the icon to the appropriate one. You can also change the icon for a station. When a station transmits to another station, a line is drawn on the map to show that connection. Throughout the program I refer to these as “connections” but in fact they may not actually be connections. It is to difficult for the program to determine if the stations are actually connected. If station A transmits to station B, they are assumed to be connected and the line is drawn. After 30 seconds, the line is changed from a heavy line to a thin line. After the time-out specified in the preferences, the line goes away. If the line is dashed, it signifies that there is more than 10% retries on that connection. After the time-out for the station, the station will go away. If you LOCK a station, it will never go away.

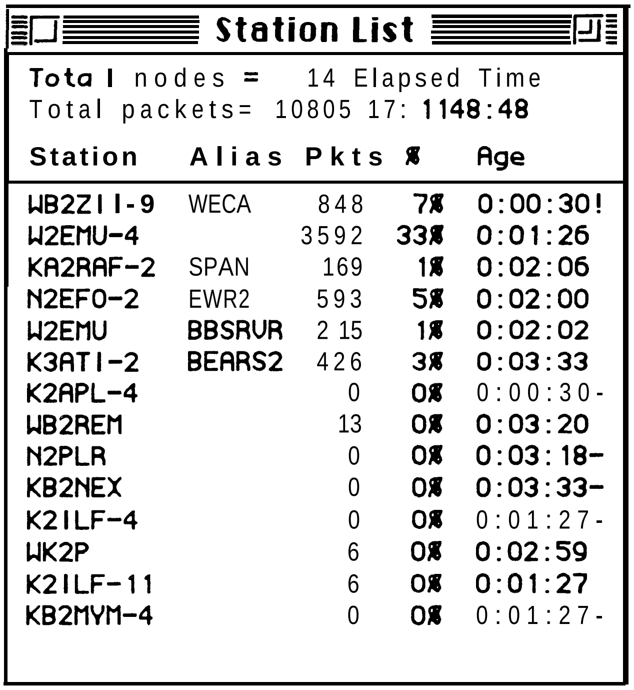


Station List Window

The Station List Window contains a list of the current active stations, it has their call sign, alias if they have one, number of packets transmitted, percentage of packets transmitted, age (hh:mm:ss) of last transmission. After the age their may be a character. The characters have the following meanings:

!	Most recent station to transmit
-	Unheard station

If you have a color screen, the station that transmitted most recently is colored red. All of the information on the Station Map window is displayed here along with some extra information such as the percentage of packets. It is just presented in manner which may be more useful in some circumstances.



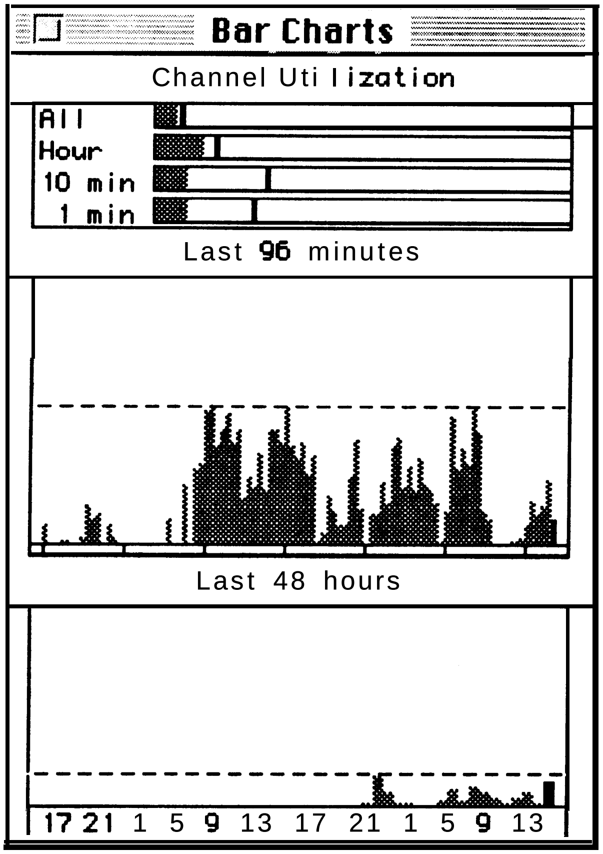
```
Total nodes =   14 Elapsed Time
Total packets= 10805 17: 1148:48

Station     Alias   Pkts  %    Age

WB2ZII-9    WECA     848   7%  0:00:30!
W2EMU-4             3592  33%  0:01:26
KA2RAF-2    SPAN     169   1%  0:02:06
N2EFO-2     EWR2     593   5%  0:02:00
W2EMU       BBSRVR   215   1%  0:02:02
K3ATI-2     BEARS2   426   3%  0:03:33
K2APL-4                0   0%  0:00:30-
WB2REM                13   0%  0:03:20
N2PLR                  0   0%  0:03:18-
KB2NEX                 0   0%  0:03:33-
K2ILF-4                0   0%  0:01:27-
WK2P                   6   0%  0:02:59
K2ILF-11               6   0%  0:01:27
KB2MYM-4               0   0%  0:01:27-
```

Bar Chart Window

The Bar Chart Window displays information about the channel utilization. The calculations are based on the number of characters received from the TNC. Therefore, the numbers are not exact but close approximations. If I implement KISS mode in a later version of this program, it will be more exact. All graphs are based on full scale being 100%



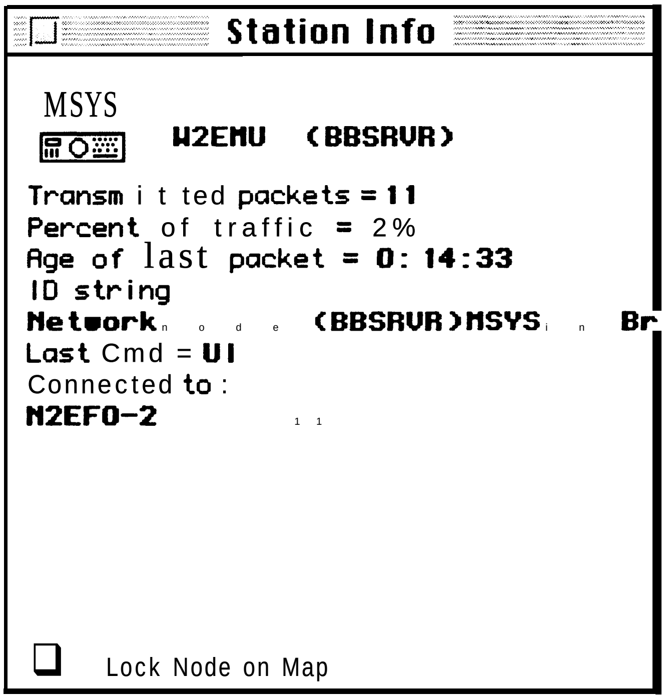
The Bar Chart Window is broken up in to 3 parts, the first part is the current utilization. It has 4 horizontal bar graphs. The first one is overall utilization since the program was started. The second is the previous hour (last 60 minutes), the third bar is the previous 10 minutes and the last one is the previous minute. The solid line in each of the bars is the high water mark.

The second graph on the window is utilization for each of the previous 96 minutes. Why 96 you ask, because it is twice 48 and I wanted 48 hours using 2 pixel wide bars for the next section and 96 1 pixel wide bars fit in the same space. The tick marks at the bottom are spaced 15 minutes apart.

The third graph is the utilization for each of the previous 48 hours. In both the last 96 minutes and last 48 hours, the most recent one is in black and the rest are in gray. The dashed line is the high water mark for both cases.

Station Info Window

To look at the detailed information about a specific station, double click on the icon in the map window or the entry in the Station List Window. This displays everything known about the station, its call, the alias if there is one, the number of packets transmitted, number of retries, age of last transmitted packet, ID string if the station has transmitted one, who the station talking to and if the station locked on the map.



```
Station Info

MSYS
W2EMU  (BBSRVR)

Transmitted packets = 11
Percent of traffic = 2%
Age of last packet = 0: 14:33
ID string
Network node (BBSRVR)MSYS in Bri
Last Cmd = UI
Connected to :
N2EFO-2        11

□ Lock Node on Map
```

The information contained in this window includes:

- The call sign of the station.
- The alias if there is one.
- Total number of packets that the station has transmitted.
- Percentage of total packet traffic
- Age of last packet transmitted.
- ID String if there was one.
- The AX.25 command string of the last packet transmitted.
- A list of the stations it is connected to. This is a list of all of the other stations that this station has transmitted to. Therefore “connected to” may not be really correct, but it is close. Also, this list does not get purged unless one of the stations itself gets purged.
  On each line of the connections, the number of packets transmitted to that station and the number of retries if any in "()".
- Lock Station on Map: If you check this box, the station will not be purged (aged out due to lack of transmission) and will be saved to the map file when you do a save command. Only those stations with this lock flag set will be saved.

Communications Window

This window is simply the raw output of the TNC as it is being received. You can watch this window to see what you were used to seeing without this program. This will show you what kinds of data is being used to create all of the maps and charts.

CONCLUSIONS

Packet Tracker has proved to be a very useful program for examining what is going on in Packet Radio. It has been downloaded hundreds of times from the various on-line services. This program has taught many of us, even some of the advanced Packet people many new things about what is going on. It has also allowed us to see where excessive retries were happening that we were not aware of before. This program has proven to be more educational than we had imagined, and has turned into a very successful program for the packet community.

Mail Tracker
A Graphical Mail Tracking Program

Keith Sproul, WU2Z
Radio Amateur Telecommunications Society

Keith Sproul, WU2Z
698 Magnolia Road
North Brunswick, NJ 08902-2647
AppleLink: Sproul. K
Internet: sproul @sproul.com
Packet: WU2Z@KB2EAR.NJ.USA

ABSTRACT

Packet Mail Forwarding as we know it today works pretty well. **[1] The** end user enters a message into the system and, other than having to address it correctly, he does not have to do anything else to make it get to its destination. If the message is addressed for some place local, it is handled via VHF only. If it is addressed for someplace far away, it gets routed to the appropriate 'gateway' to get it there. This gateway might be an HF gateway, or a SATELLITE gateway, or a WORM-HOLE, but somehow, the system knows or thinks it knows the best way for a message to get from where it is currently to where it should be. There has been much discussion on routing methods, automatic updating of routing tables, and on the best way to route messages. This can be an extremely volatile topic of discussion and will not be discussed here.

This paper discusses a network utility program called **MAIL TRACKER** that will look at messages and graphically show the route those messages have taken from the point they entered the network to their final destination. This paper will not discuss how to route traffic, only how to evaluate the routings being used.

INTRODUCTION

Mail Tracker is the second program in a series of packet utility programs specifically designed to teach us something about what is going on in our complex hobby. **Packet Tracker,** which was the first program, shows us what is going on in the immediate area on a specific frequency. It 'TRACKS' Packets and displays where they are coming from and where they are going. (See the article on this program elsewhere in these proceedings).

Mail Tracker 'TRACKS' entire Mail Messages, showing where they came from, where they have been and where they ended up. It does this by plotting the route the message took on a map. **Mail Tracker** also keeps limited statistics on several different aspects of message handling. All of this information is done in a graphical environment where a picture is worth a thousand words. In this case, telling someone that a message routing is bad is wasted breath, but showing them a picture that presents this might actually convince them that there is a problem. **Mail Tracker** runs on a Macintosh computer and since it uses data files as its input, it does not need to be connected to a TNC.

This program allows the user to select messages from the database of messages available and shows the route the message took and the amount of time the message took. The user can select a message and examine it. Once he has displayed a message he can also examine different statistics about the message and statistics about each station that message has passed through.

OBJECTIVES

There has been much discussion and arguing about message routing. There has also been lots of discussions about the content of the R-Colon lines that are added to messages as they pass through the Packet Bulletin Board Systems. In light of these 'discussions' we discussed the feasibility of writing a program to track messages that went across the country. The first obstacle was to determine if you could actually get useful information out of the data contained in the R-Colon headers that get added to the messages. Once this was determined, we had to determine how this information should be presented so that it was easily understood. After writing routines to parse all of the information and put it into databases, we started to display this information on a map. At that time, it became very apparent that this data was indeed useful and could be presented in a way to teach us something.

From this information we actually see the interesting ways that messages bounce around the country. We also can get many statistics on the different Packet BBS's around the country. The other thing that we hope to get out of this is that by showing the actual amount of travel these messages actually do, we might be able to convince some of the people to only send messages where they should be sent and not to send messages to ALLUSA or WW when they should be kept local.

MESSAGE DATA REQUIREMENTS

The data needed by this program is simply a collection of mail messages from any BBS that has the R-Colon data still intact If the messages have RFC-822 headers, they get displayed also. **Mail Tracker** does NOT provide for reading messages, just for examining the paths and the header information. All data shown in the examples in this paper were messages received at the WU2Y BBS which is fed by KB2EAR.

PROTOCOLS

Mail messages are passed from one Packet BBS to another using the WORLI / WA7MBL **[2]** protocol. Each BBS that a message passes through adds a line to the top called an R-Colon line. This line has several different variations, two of them are shown below.

```
|-------------------------------------------------------- R.:
| |------------------------------------------------------ Date/Time
I I              |--------------------------------------- BBS Call sign
I I              |     |--------------------------------- Hierarchical address
| |              |     |     |--------------------------- Additional Info
| |              |     |     |                         |-- Message BID/MID
| |              |     |     |                         |
R:930110/1159E @:WU2Y.NJ.USA [No Brunswick] MPM-Vl. 01 $:0022_KB2ICI
```

```
|-------------------------------------------------------- R.:
| |------------------------------------------------------ Date/Time
| |            |----------------------------------------- Message Number
| |            |     |----------------------------------- BBS Call sign
| |            |     |     |----------------------------- Hierarchical  address
| |            |     |     |
| |            |     |     |
| |            |     |     |
R:930115/1401z 30342@KB1BD.#CNJ.NJ.USA.NA
```

This line of data has several different forms and LOTS AND LOTS of arguments about what should and shouldn't be in it. **[3]** Another thing that is currently under hot debate is the [HANDLED BY CLOVER] message at the end of some messages. If this really needs to be included (which I don't think it should), it should be in the R-Colon header so that software like this can tell which links are CLOVER, which are HF and what other types of links are being used. It would be interesting and useful to plot the different links in different colors. This paper will not get into these arguments except to say that these R-Colon lines proved to be extremely useful, and the hierarchical addressing supplied in them made a lot of the mapping easier.

Mail Tracker uses the information contained in the R-Colon lines in tracking where the message has been. It keeps track of every BBS seen, how many messages each BBS handles, how many different countries it sees, how many different states, and how many different subareas. It also keeps track of how long a message took between stations, and the overall length of transit time. All of this is done in large database structures.

The user can look at any message in the current list and examine many different aspects of that message. The program not only displays the route taken on the map, it also displays the BBS list on the side, showing the exact BBS call sign and hierarchical address used by that BBS. Information about each message gets displayed on the bottom of the screen, The user can also select the different BBS stations and look at some of the statistics about each station.

PROGRAM USAGE

Mail Tracker displays a large window with a map of the US. **[Figure 1]** In a window at the bottom of the screen is a list of messages. The user can scroll through the list of messages, select one, and double-click on it. The data used by this program is a simple collection of mail messages. Typically, Mail Tracker will handle several hundred messages at once. This really depends on how much memory is available.

MESSAGE INFORMATION

If you double click on a message in the message list, the program displays the route the message took on the map. It also lists the BBS's the message passed through on the right hand side of the map. Some information about the message is displayed at the bottom of the map. If you double click on a BBS in the list on the right hand side of the map, it will highlight that location on the map. There are five different messages displayed in **Figures 2 through 6.**

Each message shows the message information at the top line of the bottom window. Included here are several pieces of information. The top line of Figure 2 is shown below to explain the different parts.

289	Message Number (within Mail Tracker)
WG0I	The person that sent the message
WB0GDB	His BBS
.#STP.MN.USA.NOAB	The Hierarchical address of that BBS
6/07/1993 13:05	The date and time the message was sent
R:20	The number of R-Colon headers seen
L:4	The number of LOOPS seen (See below)
T: 14 9:26	The amount of time needed, Days, Hours:Minutes

```
289 WG0I @ WB0GDB.#STP.MN.USA.NOAB 6/07/1993 13:05 R:20 L: 4 T: 14 9:26
```

On the right hand side of the map is the list of Packet BBS's the message traveled through. To follow the message, start at the BOTTOM of the list and go up. If you double click on one of these stations, it will highlight that station on the map.

STATION LOCATION

Each station that **Mail Tracker** sees gets assigned a location on the map based on the hierarchical address. The program parses this address and sees what continent, country, state, and sub-area the station is in. It then assigns a location on the map to that station based on this information.

Another way that this could be done, which is planned in future versions of this software, is to go to the call book database from CD-ROM (Buckmaster Publishing) and find the zipcode for that station. Then using another database, look up the coordinates of that zipcode. Obviously, this wouldn't work for the entire world, but would be very accurate for the US Of course, this also assumes that the call book data is up to date. Due to the size of the map and the 'rough' approximation of the sub-state area designators, the actual accuracy of the map locations are limited.

STATION STATISTICS

After selecting a message, the list of stations that it has traveled through gets displayed in the right-hand list window on the screen. The user can then select any one of these stations and bring up some statistics about that station. **[FIGURE 7]**. This includes the stations full Hierarchical address, the coordinates on the screen, the number of messages that station TRANSMITTED, and a list of all of the BBS's that it talked to and that talked to it, showing how many messages went each way. Note: This can be misleading because these statistics are ONLY collected on the messages that were received at the station where the data files were collected.

LINK STATISTICS

Mail Tracker also keeps track of how many messages pass tlhrough each LINK. A link is defined as going from station A to station B. Going from station B to station A is defined as a separate link. You can get information about the different links from any given station. **[Figure 7]**.

TIME

Mail Tracker also displays the amount of time a message took from start to end. This proved to be interesting information and brought up one minor surprise. There are a surprisingly large number of Packet BBS's that have their time incorrect. This in itself isn't a real problem, but it did prove to make some interesting things happen when examining the times and trying to determine how long messages took to pass through the system. There are many messages that arrive before they are sent out. This problem is inherent to the poor clocks on the older PCs which are quite often used in Packet Bulletin Board Systems.

WHAT IS OPTIMUM ROUTING?

The term OPTIMUM can mean different things. In this case does it mean quickest? the least amount of band-width utilization? the least amount of re-transmissions?, or the shortest number of air-miles traveled? Different people may have different opinions as to which one of these is the most important. You can never say that any one specific consideration is more important than the others all the time, but you can say what is usually more important in some 'normal' situations etc.

In consideration of message handling, most people will probably agree that fastest / shortest amount of time is either at the top or at least close to the top of importance. Using speed as a consideration, which message routes are bad and which are good? This aspect of routing takes a lot of consideration. For example, if a message that originates in Florida, what is the BEST way for it to get to the northeast? The obvious immediate answer would be straight up the eastern coast, probably through Virginia. Unfortunately, this is not the way most messages from Florida get to the northeast. Usually they go west, either to Colorado, Arizona or California, then come back east. Most people's first reaction would be that is ridiculous,, however, due to the weird propagation of' HF frequencies, it might be the QUICKEST way to get the messages from Florida to the northeast.

ROUTING CONSIDERATIONS

Another thing that has to be considered is what TYPE of message you are looking at. If the message was a private message sent to an individual at a specific BBS, it SHOULD go the shortest (quickest) route. However, if the message was a BULLETIN sent to ALLUSA, each station that handles the message asks other BBS's that it talks to if they have seen that BID number, if not, they send the message on. Using this type of forwarding, (called FLOODing) the messages can and do take very strange routes, bouncing around in non optimum routes. Is this bad? Not really, it just is confusing.

way. If this happens, the message will tend to bounce all over the place. This BID changing is something that really has to be addressed because it can cause LOTS of

will just point out that this is a problem and can cause significant amounts of unnecessary traffic.

We need to consider the different types of messages and the associated types of forwarding before we actually start to criticize a particular route.

- INDIVIDUAL / PRIVATE

- LIMITED AREA
  (NJNET, CABBS, etc.)

- WIDE AREA
  (ALLUS, WW)

As you think about how these different types of messages should be handled, you will realize that they will indeed get handled differently, even if the BBS's routing tables doesn't special-case them. Typically, messages that are aimed at local distribution do not get forwarded to the HF-gateways, or at least the HF gateway is smart enough to not send it out to its HF partners. Messages aimed at groups, i.e. limited area OR wide area will get sent to every BBS that has not seen that particular message. Messages aimed at an individual SHOULD only be sent to one other BBS. This BBS would be the BBS next closest to the final destination.

Given the fact that bulletins are usually 'FLOODed', i.e. sent to every BBS known that hasn't already seen that specific message, you can start to understand the sometimes hap-hazard routing. In this case, it is truly possible for a message that originated in Pennsylvania to make its way to New York by way of the midwest. This doesn't seem correct, but it is all done assuming that the BIDS have not been changed, and that the stations keep the BID list long enough to prevent duplication. That time needs to be at least two of three months. **Mail Tracker** has shown messages to take in excess of twenty days to travel from beginning to end.

LOOPS

Mail Tracker keeps track of something that I call a LOOP. A loop is when a message leaves from one area, i.e. a state or country, and then re-enters that state or country. In general, LOOPS show either poorly done routing tables, or confusion as to which way the message should be forwarded. When looking at the list of messages, the number labeled L: is the loop count. The higher this number, the poorer the forwarding method.

Theoretically, all loops should be bad, but there are some that are not. The best example of a loop that is good is a messages going from northwest USA such as Washington, Oregon, or Idaho to northeast USA. Usually, these messages go from USA to Canada and back to USA. Mail Tracker sees this as a loop, which indeed it is, but it is actually good, efficient routing.

CONCLUSIONS

In the process of creating the program, there were many hours of discussion as to whether the information could actually be put to good use, and was the program even worth doing. What has come out of these many discussions is that the most good would be to use this program to evaluate routes and to try to find out where the problems exist. Once that has been determined, it will be easier to actually fix them.

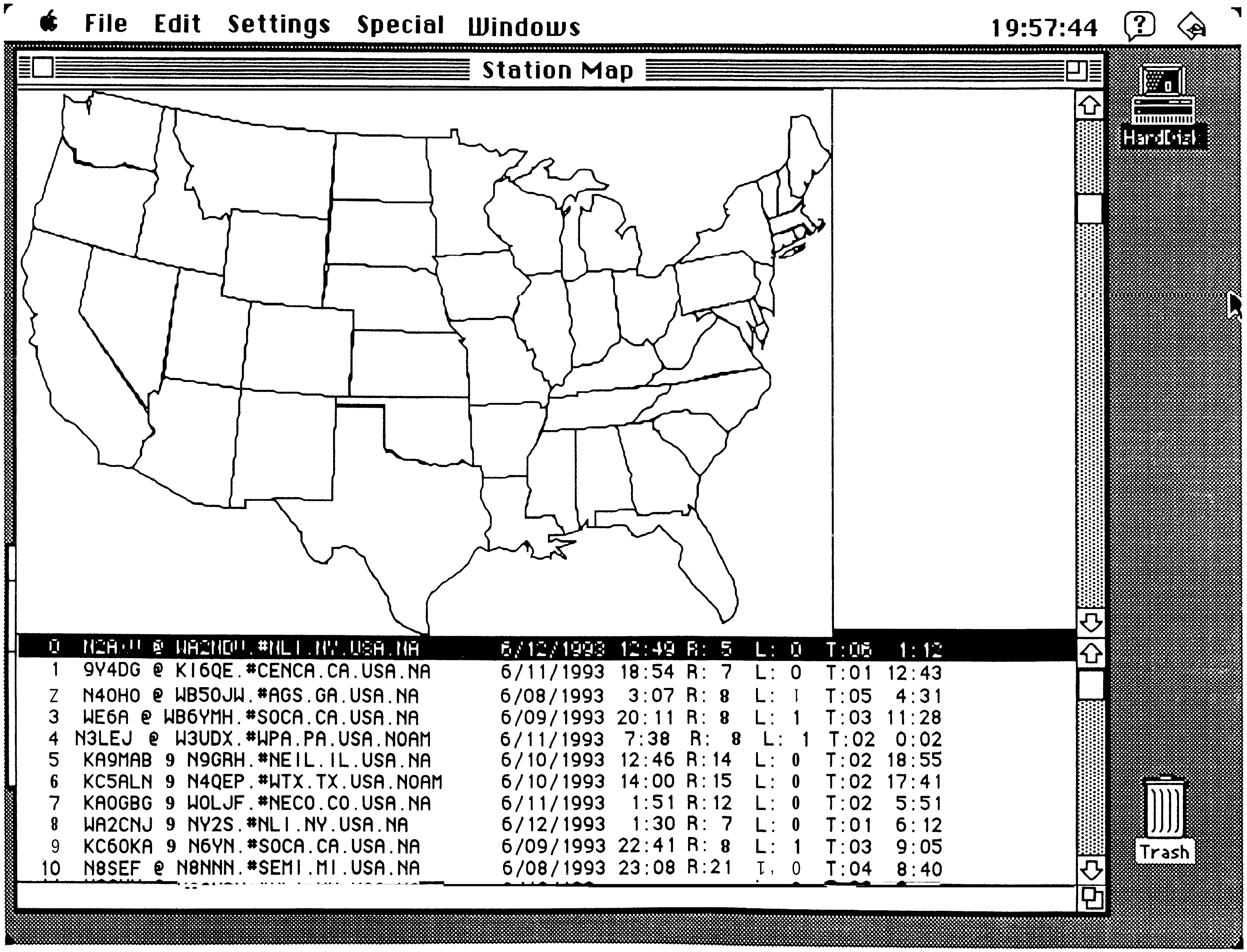
Unlike Packet Tracker, Mail Tracker is not a utility for everyone. It is really only usable by people that have access to the data files from a BBS. The information that needs to be shown to the general packet community is to show them the maps this program creates and try to convince them that sending a message to ALLUS really does have a significant impact on the Packet Mail system. These maps should be shown at ham radio clubs. If this happens, we might stop seeing messages that read TO: ALLUS, 50' TOWER, LOCAL PICKUP ONLY!!!! Yes, these types of messages still happen regularly.

REFERENCES

[1] **Judge, Ed,** W5TOO *Packet Radio's Long-Haul Mail System,* C Q Magazine, February, 1990 pp 13-17

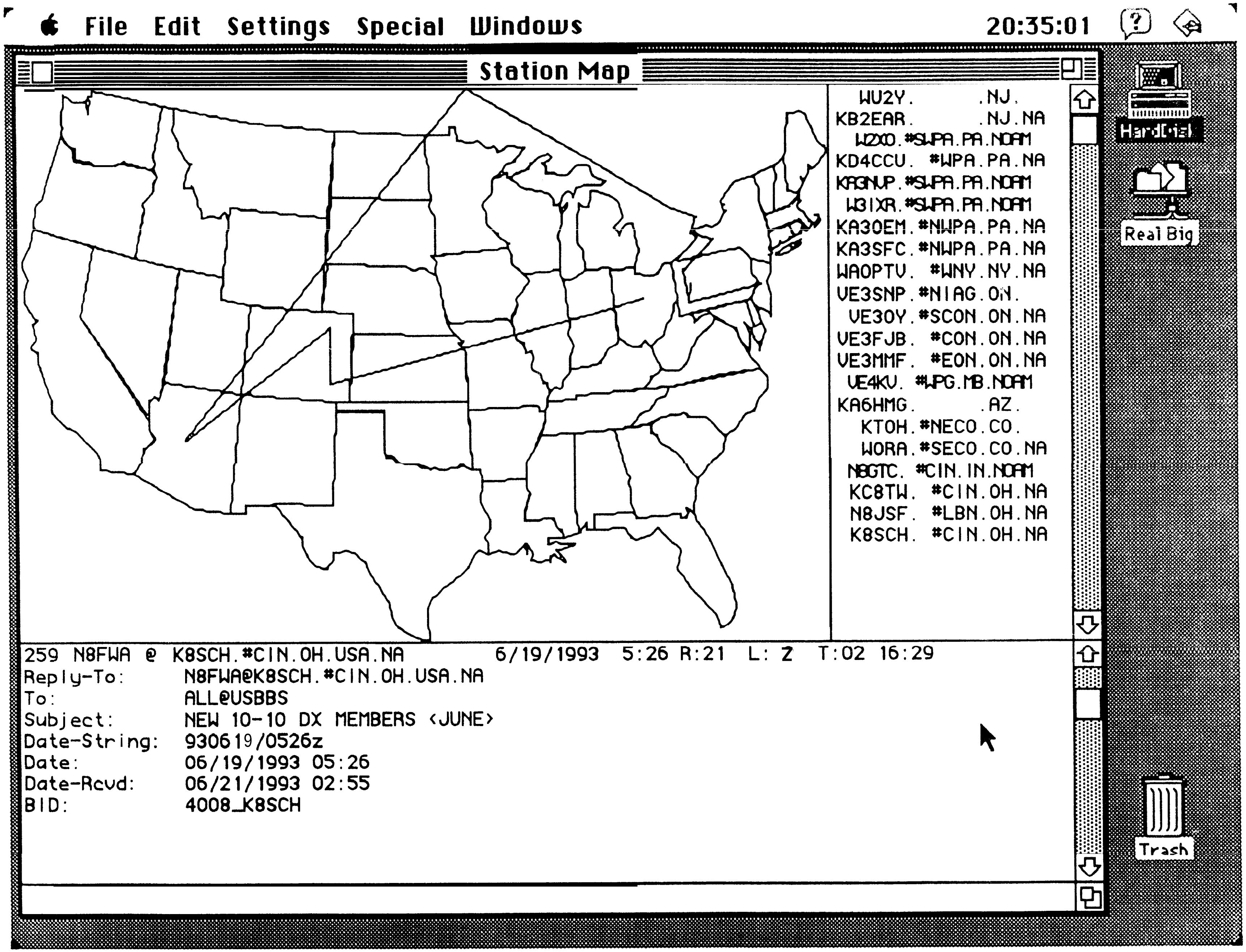
[2] **Riley, Brian B,** KA2BQE *Improving the Packet Mail Transfer System,* Proceedings of the 10th Computer Networking Conference, Newington, CT; ARRL, September 1991, pp 130- 136

[3] **Clark, Tom,** W3IWI *Some comments on the "H "ierarchical Continent Address Designator,* Proceedings of the 9th Computer Networking Conference, Newington, CT; ARRL, September 1990, pp 278-279



```
 0  N2A[illegible] @ WA2ND[illegible].#NLI.NY.USA.NA      6/12/1993 12:49 R: 5  L: 0  T:06  1:12
 1  9Y4DG @ KI6QE.#CENCA.CA.USA.NA      6/11/1993 18:54 R: 7  L: 0  T:01 12:43
 Z  N4OHO @ WB5OJW.#AGS.GA.USA.NA       6/08/1993  3:07 R: 8  L: I  T:05  4:31
 3  WE6A @ WB6YMH.#SOCA.CA.USA.NA       6/09/1993 20:11 R: 8  L: 1  T:03 11:28
 4 N3LEJ @ W3UDX.#WPA.PA.USA.NOAM       6/11/1993  7:38 R:  8 L: 1  T:02  0:02
 5  KA9MAB 9 N9GRH.#NEIL.IL.USA.NA      6/10/1993 12:46 R:14  L: 0  T:02 18:55
 6  KC5ALN 9 N4QEP.#WTX.TX.USA.NOAM     6/10/1993 14:00 R:15  L: 0  T:02 17:41
 7  KA0GBG 9 W0LJF.#NECO.CO.USA.NA      6/11/1993  1:51 R:12  L: 0  T:02  5:51
 8  WA2CNJ 9 NY2S.#NLI.NY.USA.NA        6/12/1993  1:30 R: 7  L: 0  T:01  6:12
 9  KC6OKA 9 N6YN.#SOCA.CA.USA.NA       6/09/1993 22:41 R: 8  L: 1  T:03  9:05
10  N8SEF @ N8NNN.#SEMI.MI.USA.NA       6/08/1993 23:08 R:21  T, 0  T:04  8:40
```

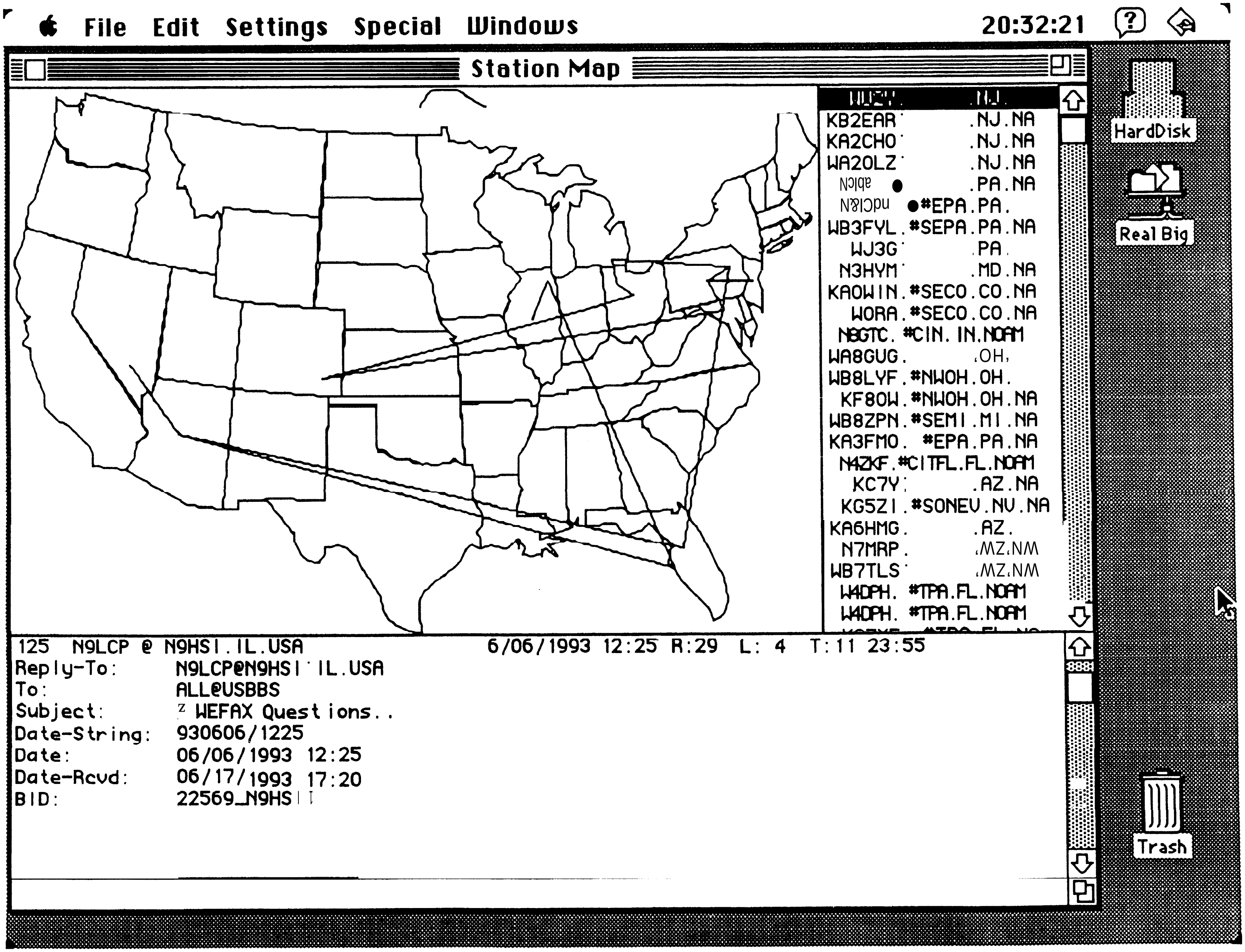
Figure 1 MAIL TRACKER MESSAGE LIST



```
WU2Y.        .NJ.
KB2EAR.      .NJ.NA
W2XO.#SWPA.PA.NOAM
KD4CCU. #WPA.PA.NA
KA3NUP.#SWPA.PA.NOAM
W3IXR.#SWPA.PA.NOAM
KA3OEM.#NWPA.PA.NA
KA3SFC.#NWPA.PA.NA
WA0PTV. #WNY.NY.NA
VE3SNP.#NIAG.ON.
VE3OY.#SCON.ON.NA
VE3FJB. #CON.ON.NA
VE3MMF. #EON.ON.NA
VE4KV. #WPG.MB.NOAM
KA6HMG.        .AZ.
KTOH.#NECO.CO.
WORA.#SECO.CO.NA
N8GTC. #CIN.IN.NOAM
KC8TW. #CIN.OH.NA
N8JSF. #LBN.OH.NA
K8SCH. #CIN.OH.NA
```

```
259 N8FWA @ K8SCH.#CIN.OH.USA.NA       6/19/1993  5:26 R:21  L: Z  T:02 16:29
Reply-To:       N8FWA@K8SCH.#CIN.OH.USA.NA
To:             ALL@USBBS
Subject:        NEW 10-10 DX MEMBERS <JUNE>
Date-String:    930619/0526z
Date:           06/19/1993 05:26
Date-Rcvd:      06/21/1993 02:55
BID:            4008_K8SCH
```

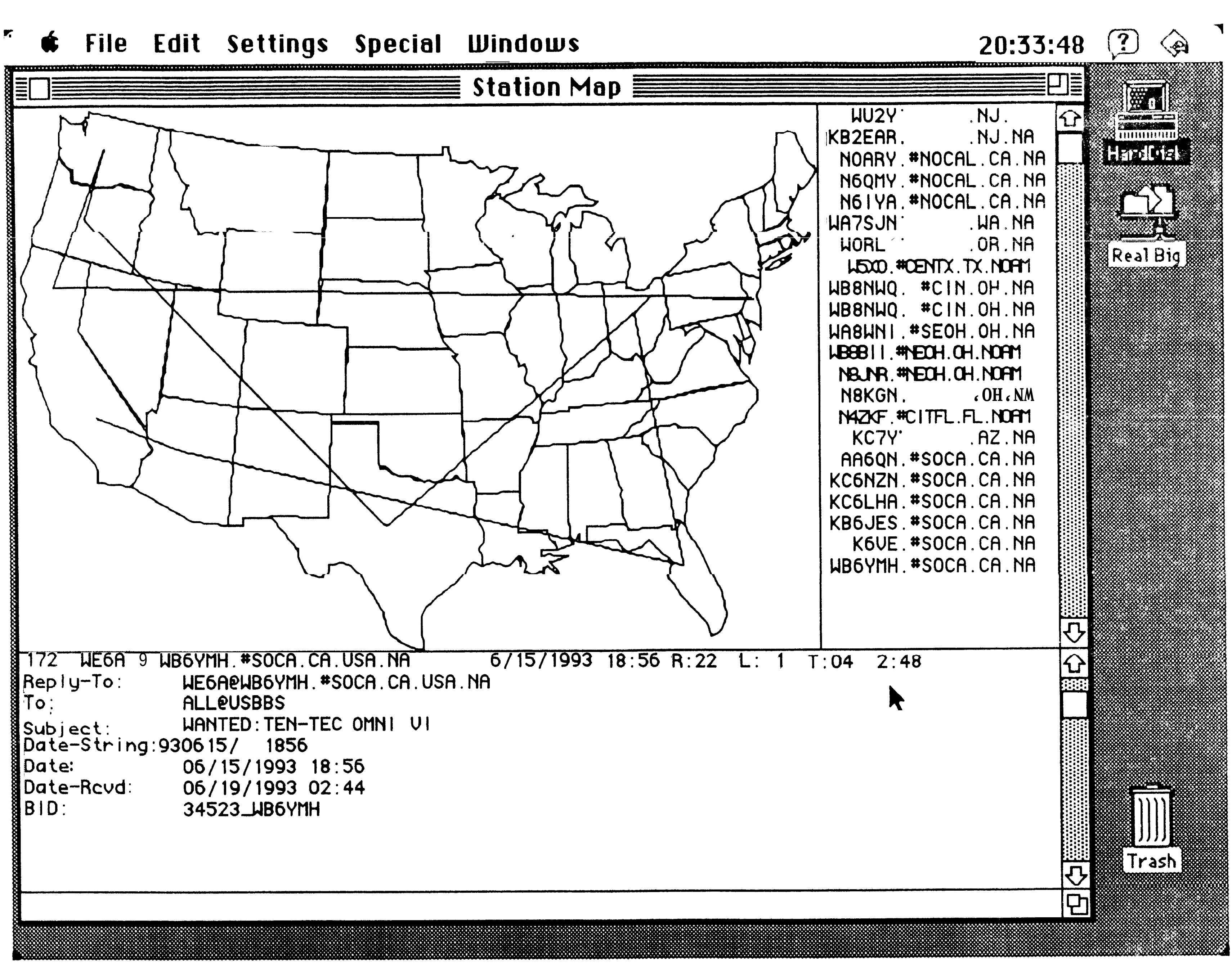
Figure 2   MAIL TRACKER MAIL PATH



```
KB2EAR      .NJ.NA
KA2CHO      .NJ.NA
WA2OLZ      .NJ.NA
N9lqe ●     .PA.NA
N9ICpu ●#EPA.PA.
WB3FYL.#SEPA.PA.NA
  WJ3G       .PA.
 N3HYM       .MD.NA
KA0WIN.#SECO.CO.NA
  W0RA.#SECO.CO.NA
 N8GTC. #CIN. IN.NOAM
WA8GUG.       .OH.
WB8LYF.#NWOH.OH.
 KF8OW.#NWOH.OH.NA
WB8ZPN.#SEMI.MI.NA
KA3FMO.  #EPA.PA.NA
 N4ZKF.#CITFL.FL.NOAM
  KC7Y;       .AZ.NA
 KG5ZI.#SONEV.NV.NA
KA6HMG.       .AZ.
 N7MRP.       .MZ.NM
WB7TLS        .MZ.NM
 W4DPH. #TPA.FL.NOAM
 W4DPH. #TPA.FL.NOAM
```

```
125  N9LCP @ N9HSI.IL.USA              6/06/1993 12:25 R:29  L: 4  T:11 23:55
Reply-To:      N9LCP@N9HSI'IL.USA
To:            ALL@USBBS
Subject:       z WEFAX Questions..
Date-String:   930606/1225
Date:          06/06/1993  12:25
Date-Rcvd:     06/17/1993  17:20
BID:           22569_N9HSI
```

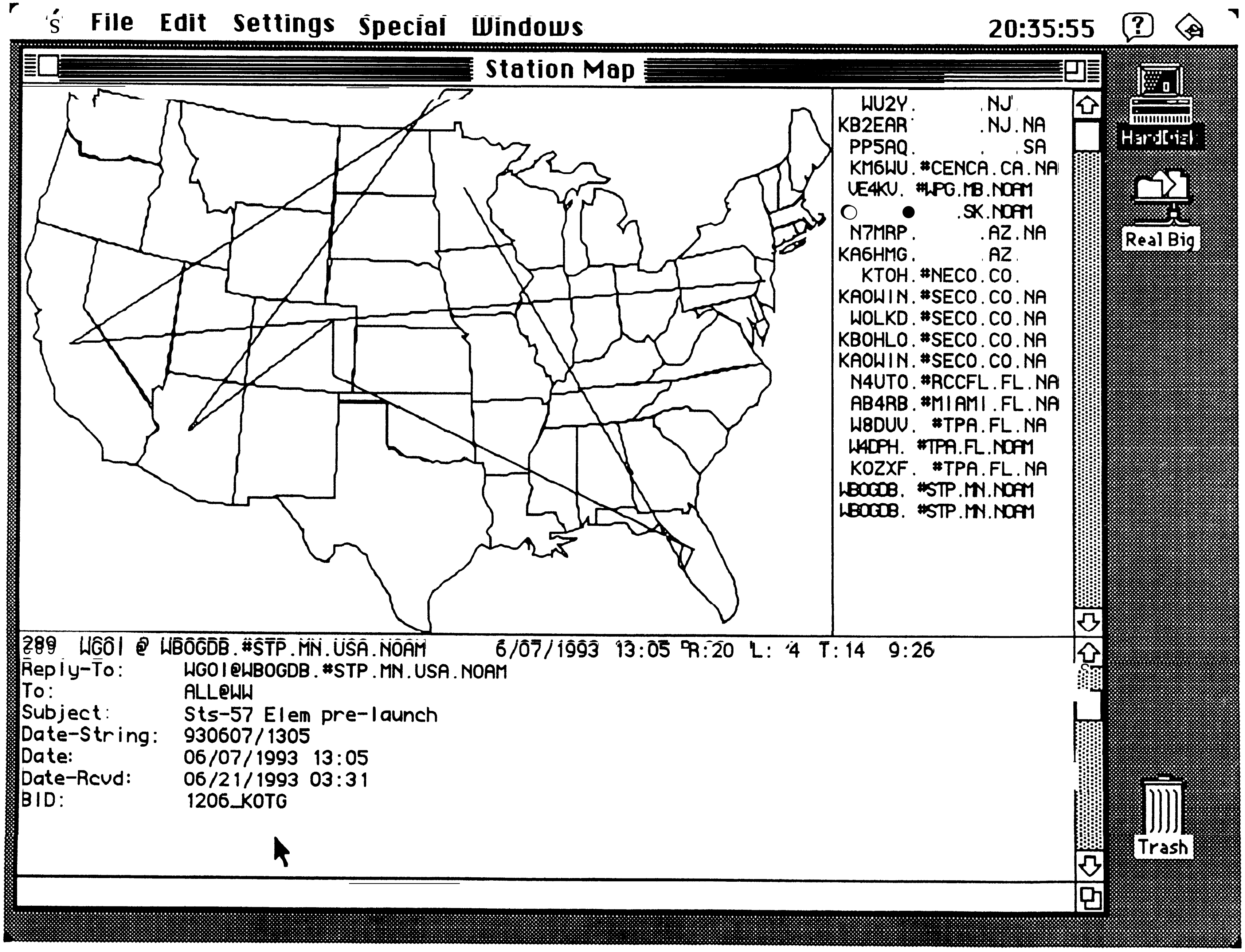
Figure 3   MAIL TRACKER MAIL PATH



```
WU2Y'        .NJ.
KB2EAR.      .NJ.NA
 N0ARY.#NOCAL.CA.NA
 N6QMY.#NOCAL.CA.NA
 N6IYA.#NOCAL.CA.NA
WA7SJN'       .WA.NA
 W0RL  '      .OR.NA
  W5XO.#CENTX.TX.NOAM
WB8NWQ. #CIN.OH.NA
WB8NWQ. #CIN.OH.NA
WA8WNI.#SEOH.OH.NA
WB8BII.#NEOH.OH.NOAM
 N8JNR.#NEOH.OH.NOAM
 N8KGN.        ,OH,NM
 N4ZKF.#CITFL.FL.NOAM
  KC7Y'        .AZ.NA
 AA6QN.#SOCA.CA.NA
KC6NZN.#SOCA.CA.NA
KC6LHA.#SOCA.CA.NA
KB6JES.#SOCA.CA.NA
  K6UE.#SOCA.CA.NA
WB6YMH.#SOCA.CA.NA
```

```
172  WE6A 9 WB6YMH.#SOCA.CA.USA.NA        6/15/1993 18:56 R:22  L: 1  T:04  2:48
Reply-To:      WE6A@WB6YMH.#SOCA.CA.USA.NA
To:            ALL@USBBS
Subject:       WANTED:TEN-TEC OMNI VI
Date-String:930615/   1856
Date:          06/15/1993 18:56
Date-Rcvd:     06/19/1993 02:44
BID:           34523_WB6YMH
```

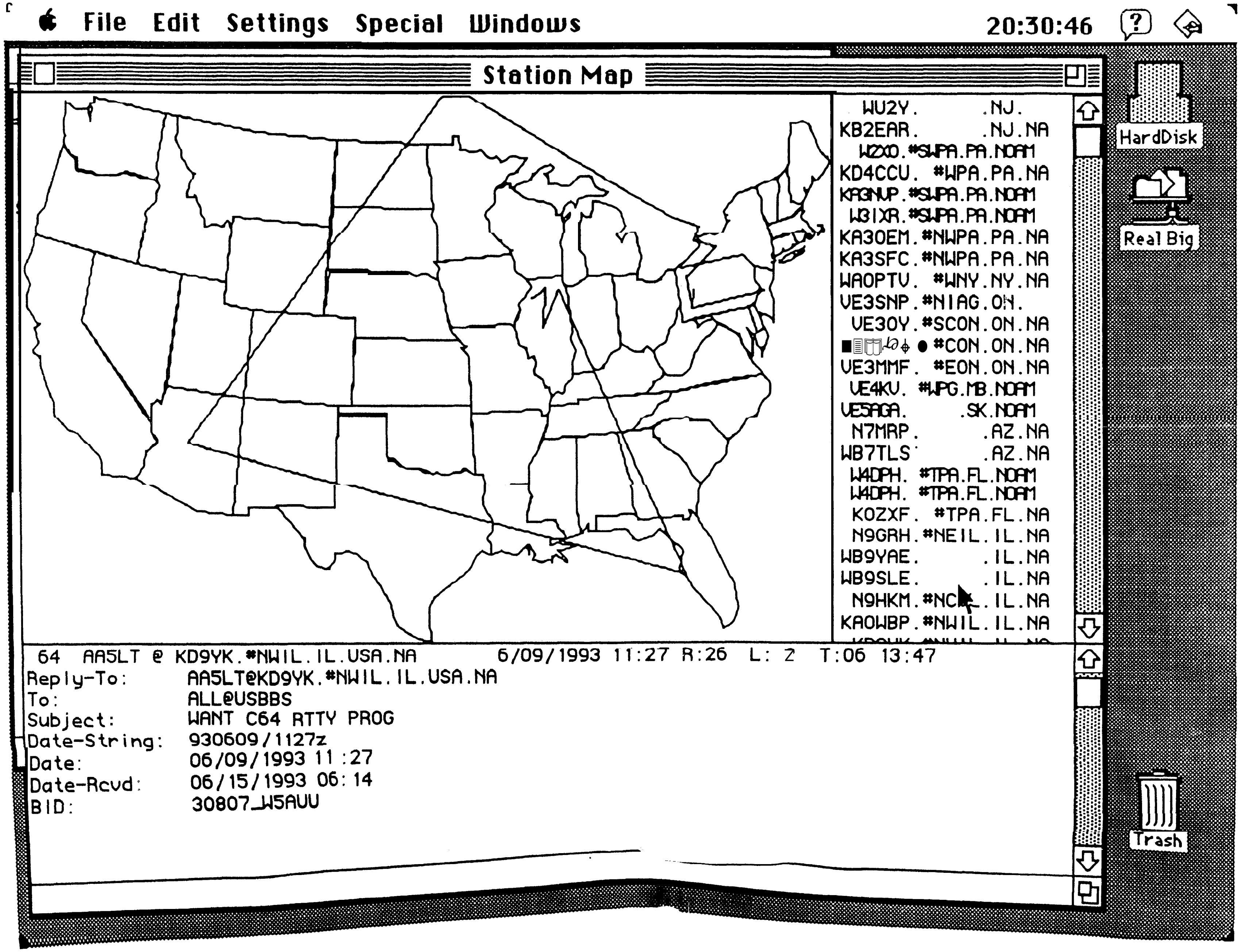
Figure 4   MAIL.TRACKER MAIL PATH



```
 WU2Y.        .NJ
KB2EAR        .NJ.NA
 PP5AQ.      .  .SA
 KM6WU.#CENCA.CA.NA
 VE4KV. #WPG.MB.NOAM
○       ●    .SK.NOAM
 N7MRP.       .AZ.NA
KA6HMG.        .AZ.
  KTOH.#NECO.CO.
KAOWIN.#SECO.CO.NA
 WOLKD.#SECO.CO.NA
KBOHLO.#SECO.CO.NA
KAOWIN.#SECO.CO.NA
 N4UTO.#RCCFL.FL.NA
 AB4RB.#MIAMI.FL.NA
 W8DUV.  #TPA.FL.NA
 W4DPH.  #TPA.FL.NOAM
 KOZXF.  #TPA.FL.NA
WBOGDB.  #STP.MN.NOAM
WBOGDB.  #STP.MN.NOAM
```

```
289  WG0I @ WBOGDB.#STP.MN.USA.NOAM      6/07/1993  13:05 R:20  L: 4  T:14   9:26
Reply-To:      WG0I@WBOGDB.#STP.MN.USA.NOAM
To:            ALL@WW
Subject:       Sts-57 Elem pre-launch
Date-String:   930607/1305
Date:          06/07/1993 13:05
Date-Rcvd:     06/21/1993 03:31
BID:           1206_KOTG
```

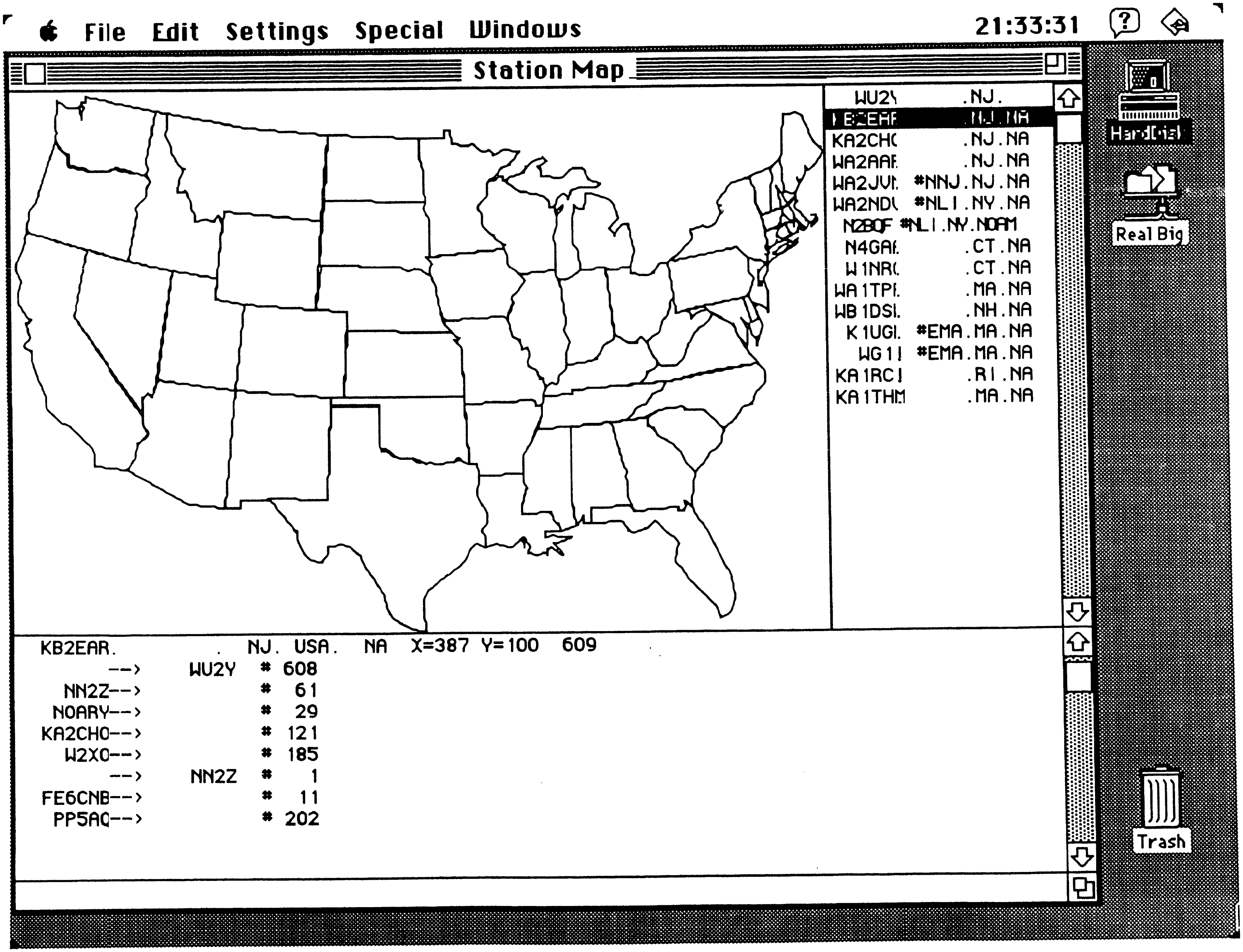
Figure 5   MAIL TRACKER MAI- PATH



```
  WU2Y.          .NJ.
KB2EAR.          .NJ.NA
  W2XO.#SWPA.PA.NOAM
KD4CCU.  #WPA.PA.NA
KA3NUP.#SWPA.PA.NOAM
 W3IXR.#SWPA.PA.NOAM
KA3OEM.#NWPA.PA.NA
KA3SFC.#NWPA.PA.NA
WAOPTV.  #WNY.NY.NA
VE3SNP.#NIAG.ON.
 VE3OY.#SCON.ON.NA
■▤⌖ ● #CON.ON.NA
VE3MMF.  #EON.ON.NA
 VE4KV. #WPG.MB.NOAM
VE5AGA.       .SK.NOAM
 N7MRP.          .AZ.NA
WB7TLS           .AZ.NA
 W4DPH.  #TPA.FL.NOAM
 W4DPH.  #TPA.FL.NOAM
 KOZXF.  #TPA.FL.NA
 N9GRH.#NEIL.IL.NA
WB9YAE.          .IL.NA
WB9SLE.          .IL.NA
 N9HKM.#NC  .IL.NA
KAOWBP.#NWIL.IL.NA
```

```
 64  AA5LT @ KD9YK.#NWIL.IL.USA.NA        6/09/1993 11:27 R:26  L: 2   T:06 13:47
Reply-To:       AA5LT@KD9YK.#NWIL.IL.USA.NA
To:             ALL@USBBS
Subject:        WANT C64 RTTY PROG
Date-String:    930609/1127z
Date:           06/09/1993 11:27
Date-Rcvd:      06/15/1993 06:14
BID:            30807_W5AUU
```

Figure 6 MAIL TRACKER MAIL PATH



```
WU2Y        .NJ.
KB2EAR      .NJ.NA
KA2CHC      .NJ.NA
WA2AAF      .NJ.NA
WA2JVI  #NNJ.NJ.NA
WA2NDU  #NLI.NY.NA
N2BOF #NLI.NY.NOAM
N4GAI       .CT.NA
W1NRC       .CT.NA
WA1TPI      .MA.NA
WB1DSI      .NH.NA
K1UGI   #EMA.MA.NA
WG1I    #EMA.MA.NA
KA1RCI      .RI.NA
KA1THM      .MA.NA
```

```
KB2EAR.         .  NJ. USA.  NA  X=387 Y=100  609
      -->   WU2Y   # 608
  NN2Z-->          #  61
 NOARY-->          #  29
KA2CHO-->          # 121
  W2XO-->          # 185
      -->   NN2Z   #   1
FE6CNB-->          #  11
 PP5AQ-->          # 202
```

Figure 7 MAIL TRACKER STATION INFORMATION

The Client/Server Bulletin Board System (csbss)

Jim Van Peursem – KEOPH
Bob Arasmith – NOARY

Abstract

The current Bulletin Board System (BBS) network is showing signs of age, There are problems with duplicate messages, new users are intimidated by their interface, the channel is becoming congested, etc. By looking at the underlying assumptions of the current systems, significant improvements can be made. Some of them require substantial changes, but the benefits far outweigh the trouble involved. This paper looks at a client/server architecture currently being developed by NOARY, KEOPH, WB6YRU, and N6ZFJ, called the csbbs (client/server bbs). The paper describes the goals of the new system, and some of the ideas being considered.

Introduction

Typical BBS's today are becoming more and more powerful. They are handling an exponentially increasing number of messages every day. They are also offering more features to their users. As compared to BBS systems just a few years ago, today's systems have added features like callsign servers, HF gateways, multiple ports, internet gateways, etc.

These advances are truly great, but because of the extra load, some other changes need to be made in order to keep the system useful and feasible into the future. Two of these changes come in the area of user interface, and channel efficiency.

User interface

BBS technology has recently been progressing in every area except the user interface. Today's BBS systems are still command line driven, just as they have always been, To use any of their features, you need to know the commands. To learn the commands, you need to read the online manual, which consumes bandwidth.

With the powerful Graphical User Interface (GUI) on today's computers, it seems silly to force the command line on all users. Why is it still done? Because that is the easiest interface to program. Besides, what other alternative is there? The BBS cannot assume any one particular architecture because there are many including Macintosh, MS-DOS, MS-Windows, C-64, Apple II, etc. Each one has it's own unique interface, and the users of these systems like to see that type of interface used. It would be best if the BBS could be separated into two pieces: one that handles all of the normal functions and message forwarding, and one that handles the user interface. This is easily done with a client/server model.

Client/Server model

The client/server model is a well known method that is used in many networking situations. It is the cornerstone for many services on intemet such as WAIS, Gopher, Archie, NFS, X-windows, etc. This is an ideal model for today's BBS systems as well. The BBS that you currently know would become the server, and a new piece, the client, would be introduced to handle the user interface. The client will run locally on the user's computer, and talk to the server at a very low level. In this way, the BBS doesn't have to worry about presenting any kind of user interface, and users can use the BBS in the comfort of their own user interface.

To better illustrate how this would work, consider the diagram in Figure 1. As you can see, the server assumes most of the responsibility of today's BBS. It forwards bulletins, stores and forwards all messages, etc. It has no interface to the user, but rather an interface to the client application. The client application is responsible for presenting the information to the user and handling user commands. Don't get this confused with the way things work now. Users won't send commands to the server. The user will interact with the client application by using the mouse and menu commands for example. Only the client application will interact with the server.

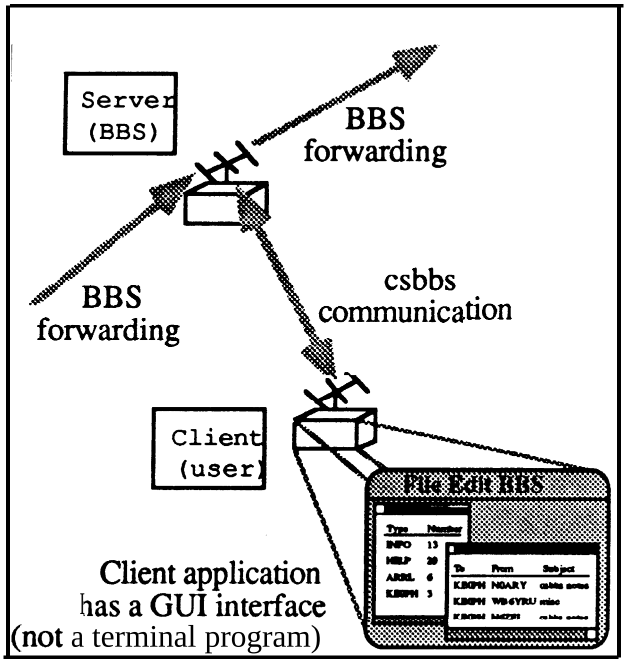


Figure 1: Client/Server illustration

When the user invokes an action that requires the server's attention, the client properly formats that request and sends it to the server. For example, if the user requests to read a bulletin (by clicking the mouse button on a message entry for example), the client will send the appropriate low level command to read it (not "r 1234"). The server will then send that message to the client in a very efficient manner. See "How to improve channel utilization" below.

The client can then store that message locally so that if the user ever wants to read it again, they can without consuming any network bandwidth. It will be listed and displayed in the same way as non-local bulletins so it will be a very familiar interface. It will be a convenient way to archive messages for later reference. This has other advantageous side effects which will be discussed later in "Message Listing".

How to improve channel utilization

Another problem with current systems is that they consume far more bandwidth than they really need to. By taking advantage of some simple techniques, the BBS's impact on the channel utilization can be changed dramatically.

Most of the utilization gains will again come from the client/server model. Since the BBS (server) isn't responsible for providing any kind of user interface, it can transfer the information in a way that only the client can decode. For example, rather than sending the message number 12345 as a text number, which takes 5 bytes, it can be sent as one 16-bit number, which takes 2 bytes. The flag bits could also fit in that same 16-bit number, for a savings of a few more bytes per message listing.

The subject, and all other text components can be compressed before transmission which would save even more. Text is very easy to compress, so a 50% savings is not unheard of. In fact, the entire body of the message would likely be sent in compressed binary to give a greater savings.

These techniques will result in some savings of channel bandwidth. Some larger savings will come because the client stores message lists locally. So for example, if a user wants to see the message listing in a different order, no transmissions are necessary. The client will use it's internal list to derive the new listing.

One more large channel utilization improvement will come from sending message listings and messages to multiple recipients at once, If N stations are receiving the information at once, a bandwidth reduction of N times is realized. This too will require some special technique because the AX.25 connection model only allows point-to-point connections [1]. Two alternatives are available.

Either the clients can snoop copies of messages being read by others, or the server can broadcast messages at the less utilized times in the day. This second method would, in effect, offer some load balancing. If users have the messages stored locally, because of the broadcasts at non-peak times, they won't need to connect to the BBS during the high congestion times.

The messages can be sent using the UI protocol, and adding internal message index numbers. For example, at the beginning of sending the body of a message, first send the BID string. This will uniquely identify this message. Then number every frame sent so that a client knows if it has missed a frame. When this situation arises it can either send a request to immediately fill this frame, or wait until the next time the message is sent and grab a copy of the missing frame.

In general, it seems appropriate for the server to send out messages at periodic times throughout the day. This would be done in an effort to maintain some sort of channel load balancing. If messages can be sent from server to clients during the very light utilization times, it will help ease the heavy usage times. This will require a lot of experimentation to determine if it is feasible or not. It might also border on some legal issues. This may be determined to be broadcasting since it is a simplex operation. The term, "dispersion" of information is preferred.

Initially, the message snooping idea seems more feasible. Since there are so many new messages arriving at the BBS's every day, it will take a lot of dispersion for them to get to everyone. However, the more popular messages are read multiple times by users, so the possibility of successfully snooping a copy of them is very likely. The message listing can also be snooped this way. In fact, the clients may even be able to snoop messages during BBS forwarding too.

Server Arbitration

Since BBS's are a. hot spot for many users, they are a cause for great network congestion. While many stations are competing to communicate with them, there is a great potential for collisions. Some sort of arbitration scheme may help out.

Some areas in Germany are using DAMA [2] as an arbitration mechanism for their node stations. We will be experimenting with this technique and probably others in an effort to reduce the hidden transmitter problem and collisions in general.

Message Listings

Another area where BBS systems have not shown much progress is in the area of viewing message listings. The standard viewing technique is still chronological. The listing goes from the most recently received message, to the least recently received message. This is a very cumbersome way for users to view messages. What about viewing them by subject or who they are addressed to or from?

Hams are starting to address their bulletins in a more useful way, such as using the TO: field to give the general category of the bulletin. This is very useful and should be taken advantage of better. The user should be able to view messages by categories. The user should also be able to search the subject field for certain words, or use regular expression searches. NOARY currently provides these powerful capabilities in his BBS software, and they are very popular.

All of these things can be done very efficiently now because the message listing will only be sent from the server to the client once. The client will store the listing and present it to the user in any way that they want without ever using any network bandwidth, because it is totally a local operation. If the user wants to see a different subset or see the messages in a different order, the client application takes care of this request locally.

Conclusion

The client/server idea for BBS's will not take over immediately, but it will be fairly simple to use the existing BBS systems along with the csbbs systems. In fact, current BBS systems can set up another SSID port that acts as the server port. That way, those users that don't have a client application can still use the BBS in the way that it is used today. The BBS and server parts can share the same internal storage and functionality.

The project is currently in the early coding stage. We are still debating many issues. We hope to have some very early results soon. NOARY will be adding the server protocol to his BBS software, and others will be working on client applications for different computers.

We are trying to make the interface between the client and server as general as possible so that it will work for all computer systems. We currently have representatives working on client applications for the Macintosh (KEOPH), MS-DOS (WB6YRU), and MS-Windows (N6ZFJ) architectures. We would like to see the C-64, Apple II, etc. represented as well so that we don't inadvertently make any bad assumptions. If you would like to volunteer, contact NOARY for more details.

Acknowledgments

Many thanks to NOARY for heading this great idea. He has an outstanding BBS program that will be used as the starting point for the server application. Thanks also to the people currently working on the client applications. For more information, feel free to contact them directly.

Bob Arasmith – NOARY – Server
837 Jasmine Dr.
Sunnyvale, CA 94086
NOARY @ NOARY.#NOCAL.CA.USA
internet: n0ary@arasmith.com

Jim Van Peursem – KEOPH – Macintosh
RR#1, Box 83A
Kelley, IA 50134
KEOPH @ NOAN.IA.USA
intemet: jvp@iastate.edu

Gary Mitchell – WB6YRU -MS-DOS
185 1 Laurinda Dr.
San Jose, CA 95124
WB6YRU @ NOARY.#NOCAL.CA.USA

Connie Arasmith – N6ZFJ – MS-Windows
837 Jasmine Dr.
Sunnyvale, CA 94086
N6ZFJ @ NOARY .#NOCAL.CA.USA

References

[1] T.L. Fox – WB4JFI, *AX.25 Amateur Packet-Radio Link-Layer Protocol. Version 2 .O* Newington, CT: ARRL, 1984.

[2] Detlef J. Schmidt – DK4EG, *DAMA – A New Method of Handling Packets?* ARRL Amateur Radio 8th Computer Networking Conference, 1989, pg 203–209.

RMAILER: The Next Generation

Frank Warren, Jr, KB4CYC
Radio Amateur Telecommunications Society
114 Westervelt Avenue #23
North Plainfield, NJ 07060-4162

At the Eleventh Computer Networking Conference, RMAILER an add-on server for the remote ad hoc mailing list protocol known as RMAIL (Remote MAILer) was introduced. RMAILER has evolved since that time taking some at times interesting turns in development.

1. Background

RMAIL briefly works as follows: a single **RMAIL** message, including addressing information for the intended recipients, will traverse the network to a point closer to final delivery. Upon reaching that point, the single message will expand into multiple messages, one for each named recipient.

Those wishing further details on the RMAIL protocol should refer to the Author's previous paper.

At the time of the original paper, the only "RMAIL aware" centralized distribution server was **ROSEDIST,** then an integral part of the ROSErver/PRMBS PBBS package now ironically replaced by **RMAILER.**

2. Developments Over the Past Year

In the current release, **RMAILER** supports the use of **RFC-822** style continuation lines for the **"To:"** line, as envisioned at the time the original **RMAIL** paper was published.

The intended companion centralized distribution list server was never developed as a separate program. Instead, it became a trivial alternate case of providing for continuation lines. The development of this "Next Generation" RMAILER caused a distinct easing of workload for a number of sysops in the Northeastern US as yet another feature, the archiving of "R:" headers to the log upon expansion and re-origination of messages, reduced the load from apparent duplicate messages generated by the Star Trek discussion list (TREKML@KB4CYC.NJ.USA.NA) the author hosts.

The most dramatic performance improvement of the "Next Generation" RMAILER is the ability to process embedded RMAIL, RMAILs composed of other RMAILs, allowing multiple levels of nesting for optimizing distribution lists.

In addition to being a direct optional replacement for the ROSErver/PRMBS RMAILER.EXE distributed with releases 1.79 and later (code derived from the author's 2.03/2.04 release), the current RMAILER can act as a "sewer" under the F6FBB BBS system when the .EXE file is given a name that includes "FBB" (e.g., FBBRMAIL.EXE). Used in the ROSErver/PRMBS environment, **RMAILER** also provides loop indication avoidance re-addressing and Address Correction Notification services.

The most recent releases of RMAILER add maintenance services for "fixed" mailing lists (e.g., TREKML or ASKRAT) by automatically processing subscription requests (i.e., ADD, SUBSCRIBE, DELETE or UNSUBSCRIBE) in addition to list content requests **(LIST,** SENDLIST) all via the subject line. List maintainers are given control of how open a list they choose to operate through a set of "*" directives **(*CONFIRM, *NOLIST** and ***RESTRICT)** included in the list.

3. Creation of Embedded RMAIL

Embedded RMAIL messages use the inner RMAIL@<bbs> address and a pseudo-address of ***EOF to set off the included list. If the embedded list includes all remaining addresses the ***EOF may be omitted. An example would be:

```
To: rmail@kb4cyc, kb4cyc, rmail@kb7uv, kb7uv, rmail@wb2gtx, ka2usu,
                n2irz, ***EOF, wb2ibo@wb2ibo
```

This message will first go to KB4CYC where a copy is dropped for KB4CYC. **A** message to RMAIL@KB7UV including the remainder of the list is created. At KB7UV messages to KB7UV, RMAIL@WB2GTX and WB2IBO@WB2IBO are created and processed as needed. The RMAIL@WB2GTX will expand on arrival there for KA2USU and N2IRZ.

4. Distribution (Mailing List) Files

The distribution list files (<listName> .DST ex., askrat.dst) contain lists of addresses, one per line, plus any needed "*" directives. The files may also contain comments either in lines begining with a pound sign ("#") or following addresses seperated by white space (space or tab). **A** distribution file that would produce the embedded **RMAIL** above might read **as** follows:

```
# SAMPLE  Distribution
*CONFIRM  Send confirmation to message originator
rmail@kb4cyc
kb4cyc
rmail@kb7uv
kb7uv
rmail@wb2gtx
ka2usu
n2irz
***EOF
wb2ibo@wb2ibo
***EOF
# End of SAMPLE list
```

The type of a non-RMAIL message created by a distribution list may be forced by preceeding the address with either "p/" or "b/" (ex., p/kb4cyc or b/sample@njnet).

5. Distribution

RMAILER is available at no charge for non-commercial use within the Amateur Radio, MARS, RACES, and CAP services.

RMAILER is distributed as a PK Zip-ed archive RMAILxxx.ZIP (where xxx [currently 213] is the version number * 100) containing files RMAILER.EXE (the executable) and RMAILER.MAN (a UNIX style “manual page”) in addition to other useful utilities by the author.

RMAILER can be downloaded from CompuServe HamNet Forum, the KB7UV Landline ROSErver/PRMBS (see below), “HIRAM” (the ARRL multi-user telephone BBS), and other telephone BBSs.

Those desiring RMAILER on MS-DOS magnetic media should send pre-formatted diskettes and return postage to the Radio Amateur Telecommunications Society (see below).

Requests for the code via the Internet should be directed to kb4cyc@kb2ear.ampr.org.

The author may be contacted via packet as kb4cyc@kb4cyc.nj.usa.na.

6. The RATS Open Systems Environment

RMAILER is an element of the RATS Open Systems Environment (ROSE), a project of the Radio Amateur Telecommunications Society (RATS).

Other elements of ROSE include the ROSE X.25 Packet Switch by Tom Moulton, W2VY; the ROSE/OCS On-line Callbook Server by Keith Sproul, WU2Z, and Mark Sproul, KB2ICI; ROSErver/PRMBS, the Packet Radio MailBox System by Brian Riley, KA2BQE; and STS Station Traffic System by Frank Warren, KB4CYC.

Correspondence may be sent to:

RATS
PO Box 93
Park Ridge, NJ 07656-0093

The RATS KB7UV Landline ROSErver/PRMBS supports data rates of 1200 to 14400 bps (V.32bis, V.42bis), and J-, X-, Y-, and Zmodem binary protocols. It can be reached at 718-956-7133.

7. Acknowledgments

RMAILER still would not be possible were it not for the pioneering RMAIL development included within the ROSErver/PRMBS PBBS package by Brian Riley, KA2BQE.

Referenced Trademarks

PK Zip is a trademark of PKWare, Inc.
Star Trek is a registered trademark of Paramount Pictures Corp.
UNIX is a registered trademark of UNIX Systems Laboratories, Inc.
MS-DOS is a registered trademark of Microsoft Corporation.
CompuServe is a trademark of CompuServe, Inc.

SoftWire

Aaron Wohl - N3LIW

SoftWire is a system for designed schematics using digital logic. The schematics are translated into assembly code for a target microprocessor.

A friend in the hospital had been a radio man in WW II. I built him a get well card with a single chip cpu that played a get well message. Later, I changed the program to become a useful tool to myself. The program I inserted was the Morse code alphabet which changed my get well message into a single chip morse code tutor. It became an easy learning tool for me as well as a conversation piece.

Several months later I came across an article in a ham magazine for a much simpler circuit that needed four TTL chips. Why did they use four chips -- when one would do?

I asked local hams how they designed and built logic/controller circuits. All but one used TTL chips and their general census was any kind of assembly programming was beyond them. However, the individual ham operator attempting to program a microcontroller was genuinely interested in/with the idea but unable to get anywhere. He felt his lack of progress was due to the fact that ham radio exams and study manuals often speak of logic gates but do not mention microcontroller instructions.

SoftWire is a digital logic schematic drawing program. It translates the schematic into assembly instructions for a target microprocessor or PAL assembler. SoftWire is currently under development and is being written in Smalltalk/V for Microsoft Windows. The initial target architecture is the Microchip PIC16Cxx series.

SoftWire provides icons for:

- Traditional logic blocks, AND, **OR, flip-flops** (D, RS, JK)
- Finite state machines
- Time meters to measure delays through parts of the schematic.

SoftWire schematics can be annotated with a 'time meter'. SoftWire updates this display to indicate the delay a signal experiences from one part of the circuit to another. For example:

- An output pin must be updated 20μs after changes in any of the input pins. Then a meter icon is connected to the input pins and output pins. It is set to read in micro seconds where the meter then is updated after code generation.

- This is analogous to the way word processors update page break indicators after printing or repaginating.

- If the delay shown on a meter is larger than real time constraints allow, a constraint may be set on the meter to ask for a delay less than or equal to a given time. The SoftWire compiler will try harder to speed up that path, perhaps at the cost of increased code size.

  Constraints may also be set for minimum and maximum delay so as to produce a constant response time for a path.

- The meter metaphor provides a powerful tool to deal with real time constraints. Traditional high level compilers have optimization settings on a per file basis. However, they cannot express specific constraints for certain paths to be fast enough, and cannot express the concept of making some execution paths arbitrarily slow.

The SoftWire schematic as a visual program metaphor is attractive for logic replacement for end users and is more hardware oriented. Some problems however, still remain. SoftWire currently thinks of each 'wire' in the schematic as carrying a bit from a single output to some number of inputs.

- A tri-state IO pin icon currently has an input connector dot, an output connector dot, and a direction connector dot. This does not look very natural in reading a schematic.

- The Microchip PIC16C71 has an onboard analog to digital converter. It would be more natural to deal with the output of the converter as an eight bit integer flowing through a pipe rather than as eight wires.

The SoftWire program is available from the CompuServ ham radio sig, software library. It is also available on the internet from host akutaktak.andrew.cmu.edu [128.2.35.1] in the /aw0g directory. The author can be contacted via internet mail as n3liw+ @cmu.edu.